LEVEL 2

A SYNTACTIC METHOD FOR TIME-VARYING PATTERN ANALYSIS

Tzu-I J. Fan
K.S. Fu

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

TR-EE 81-10
May 1981

This work was supported by the ONR Grant N00014-79-C-0574

85

6 A SYNTACTIC METHOD FOR TIME-VARYING PATTERN ANALYSIS

10 Tzu-I J. Fan ~~and~~ K. ~~H.~~ Fu
School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

12 145

14

TR-EE 81-10

11 May 1981

15

This work was supported by the ONR Grant N00014-79-C-0574

292000

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Professor K. S. Fu for his guidance, advice and support during the research for this thesis. I also wish to thank my committee members Professor K. Fukunaga, Professor K. Hwang, and Professor G. Campbell for their suggestions.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

A syntactic method for the analysis of time-varying image patterns is proposed and studied. This method utilizes translation schema to model the time-varying properties of image patterns. A syntactic deformation model is first applied to transform the i-th image into the (i+1)-th image of an image sequence. Then the concept of translation in formal language theory is used as a mechanism to characterize the dynamic process of the image sequence. A formulation of stochastic translation is also presented. A generalized syntax-directed tree translation model is proposed to handle high-dimensional patterns. The generalized model is compared with the conventional top-down and ottom-up tree translation models.

A traffic monitoring problem is analyzed using the proposed tree translation model. Each input image is represented as a tree structure. he proposed tree translation model is used to model the variation of image content between consecutive images. A parsing algorithm for tree translation is applied to match moving objects (vehicles) in each pair of consecutive images.

CHAPTER 1

INTRODUCTION

1.1 Introduction

During the past two decades, there has been an increasing interest in pattern recognition. Most of the developments in the theory and applications of pattern recognition use the statistical approach [1-6]. In order to represent the structural information contained in the patterns, the syntactic or structural approach has been proposed [7-10]. The precision of syntactic specification provides the recognition procedure not only the capability of classifying patterns but also the capacity of describing patterns. Recently, the problem of time-varying image analysis has drawn great attention [15-20]. In this research, this problem will be analyzed using syntactic method.

A syntactic pattern recognition system consists of 3 major parts: (a) preprocessing and segmentation, (b) primitive extraction, and (c) syntax analysis. To analyze time-varying images, first of all, we need to select a proper representation method. Then a syntax analysis scheme has to be devised.

In the past, string grammar has been applied to the problem of shape analysis and waveform analysis, tree grammar has been applied in the problem of texture analysis, fingerprint recognition, and scene analysis. What's needed here is some description method which can be applied to represent and analyze general time-varying image patterns.

1.2 Survey

1.2.1 Pattern recognition

The many different mathematical techniques used to solve pattern recognition problems may be grouped into two general approaches; namely, the decision-theoretic approach and the syntactic approach. In the decision-theoretic approach, a set of characteristic measurements, called features, are extracted from the patterns; the recognition of each pattern is usually made by partitioning the feature space [1]. Most of the developments in pattern recognition research during the past fifteen years deal with the decision-theoretic approach and its applications [1-6]. In some pattern recognition problems, the structural information which describes each pattern is important, and the recognition process includes not only the capability of assigning the pattern to a particular class, but also the capacity to describe aspects of the pattern that make it ineligible for assignment to another class. A typical example of this class of recognition problem is picture recognition. In this class of recognition problem, the patterns under consideration are usually quite complex and the number of features required is often very large, which makes the idea of describing a complex pattern in terms of a composition of simpler subpatterns very attractive. In order to represent the hierarchical structural information of each pattern, that is, a pattern described in terms of simpler subpatterns and each simpler subpattern again described in terms of even simpler subpatterns, etc., the syntactic approach has been proposed [7-10]. This approach draws an analogy between the structure of patterns and the syntax of languages. Patterns are specified as being built up out of subpatterns in various ways of composition, just as phrases and sentences are built up by concatenating words, and words are built up by concatenating characters. The simplest subpatterns are called pattern primitives. The language that provides the structural description of patterns in terms of a set of pattern primitives and their composition operations is sometimes called the pattern description language. The rules governing the composition of primitives into patterns are usually specified by the so-called grammar of the pattern description language.

A syntactic pattern recognition system can be considered as consisting of three major parts, namely, preprocessing, pattern description or representation, and syntax analysis (Figure 1.1). The functions of preprocessing include (i) pattern encoding and approximation; and (ii) filtering, restoration and enhancement.

An input pattern is first coded or approximated by some convenient form for further processing. Techniques of filtering, restoration or enhancement are used to clear the noise and improve the quality of the coded patterns. At the output of the preprocessor, presumably, we have patterns of reasonably "good quality". Each preprocessed pattern is then represented by a language-like structure. This pattern representation process consists of pattern segmentation and primitive extraction. In order to represent a pattern in terms of its subpatterns, we must segmentize the pattern and, in the meantime, identify the primitives in it. In other words, each preprocessed pattern is segmentized into

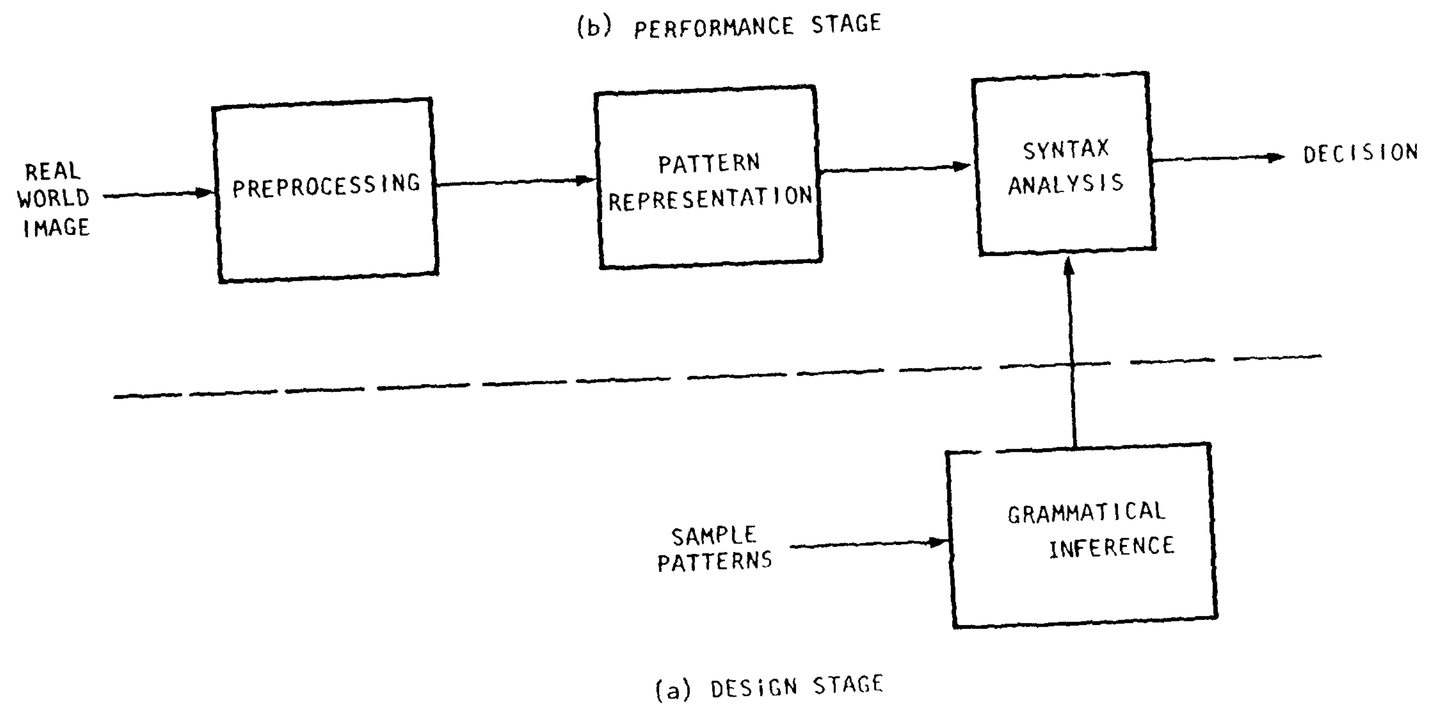


Figure 1.1 A syntactic pattern recognition system

subpatterns and pattern primitives based on prespecified syntactic or composition operations, and, in turn, each subpattern is identified with a given set of pattern primitives. At this point, each pattern is represented by a set of primitives with specified syntactic operations. The decision whether or not the representation is syntactically correct will be made by the "syntax analyzer" or "parser". When performing the syntax analysis or parsing, the analyzer can usually produce a complete syntactic description, in terms of a parsing tree, of the pattern, provided the latter is syntactically correct.

For proper representation of pattern structures, different languages has been proposed. String language has been applied in the problem of shape analysis and waveform analysis (Figure 1.2). Tree language has been applied in the problem of texture analysis, fingerprint recognition and scene analysis (Figure 1.3). Recently attributed grammar is proposed and PEE (primitive-extraction-embedded) parser is designed to increase classification performance [13].

1.2.2 Time-varying image

The various methods applied to the analysis of time-varying images can be classified into two groups. For research in the first group (cross-correlation technique and image differencing technique), analysis operations are conducted at image pixel level directly, while for the second group, actual analysis starts after some features (boundary, etc.) have been extracted from each individual image in the sequence. These techniques are further described as follows:

Cross-correlation technique [15,18,22,41,64,65,66]: Leese et al. [18] give a typical example. They compare two successive pictures with

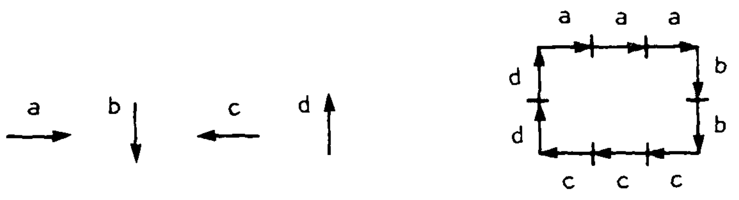


Figure 1.2 A rectangle and its pattern primitives

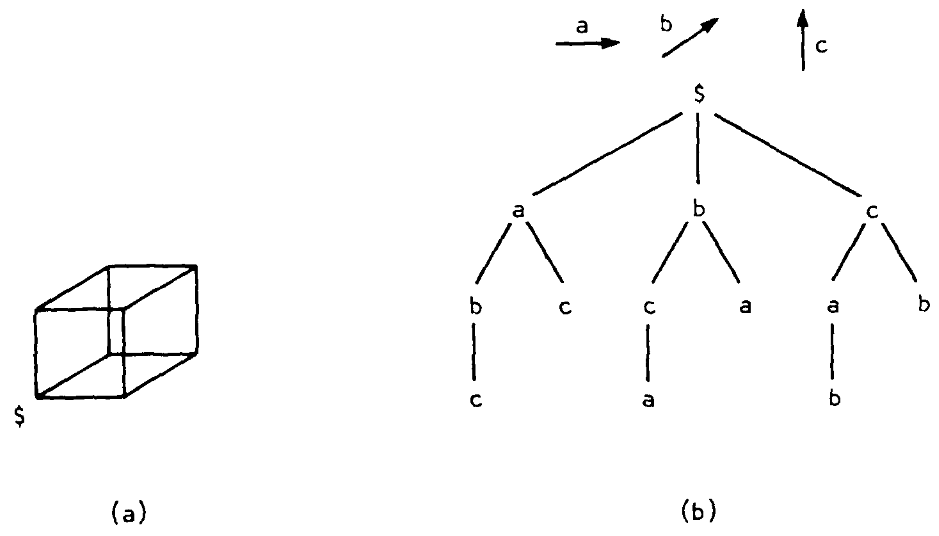


Figure 1.3 Tree representation of a square object

the first picture being divided into systematic sections (64x64 pixels). Then for each section, some reasonable area of the second picture is searched for a good match to the original section. They form a cross-correlation coefficient using the fast Fourier transform on the full gray values within the section. The cross-correlation coefficient is computed for each pairing of the original section with a candidate section in the second picture. The candidate section which yields the maximum coefficient is chosen as the match. Then a motion vector is computed as the distance and direction between the center of the original section and the center of the match section. The motion vector is essentially assigned to the section, not to any object within the section.

Image differencing technique [23,30,31,46,47,67,68,69,70]: In [46,47], the image differencing technique is applied to find the variation between two images of the same scene. The images are carefully aligned by both spatial coordinates and gray value. The spatial registration is done by considering one image as the reference image and then distorting the other image until they are aligned. The distortion is a localized procedure which operates on subregions of the images. Cross-correlation technique is used to compute the amount of distortion necessary to align a subregion with its corresponding subregion in the reference image. After the spatial registration has been completed, a point-to-point subtraction process generates a third image which displays the variation between the given images.

Limb and Murphy [67] report a hardware implementation of subtractive method. In addition to the subtraction process between the corresponding pixels of two consecutive images, within-image comparisons are made among pixels and their suitable neighbors. The within-image comparisons are used to normalize the between-image comparisons. These comparisons are essentially the absolute differences of the pixel gray values and are summed over the entire image. This yields a velocity estimate for the image as a whole and not for any specific object.

The research work in group II includes [37,39,42,43,48,49,72,73,74,77]: Chow and Aggarwal [43] develop an algorithm to track moving objects in an image sequence. After preprocessing of an image, they build a model which consists of the objects extracted from the image. Each object is represented as a set of feature values: area, centroid position, etc. The matching process of moving objects between consecutive images is performed through the comparison of these feature values. After the matching, the model is then updated with new information. The analyzed results include the number of moving objects and their corresponding velocities in the whole sequence. Other similar works include [37,39,42,49,77].

Recently O'Rourke and Badler report a system for the analysis of human motion images [73]. The model for human body is composed of segments and joints. A joint is a unique point connecting two segments. A segment is an abstract rigid body with an associated embedded coordinate system. The surface of each segment is defined by a collection of graphical primitives, called spheres, located at fixed positions within the segment's coordinate system. There are constraints governing the relations between different parts of the body. These constraints arise from the structure of the body, gravity effect, etc.

Based on the analysis of the past images, the system forms a motion description which is comprised of a set of piecewise linear functions. Each linear function describes the location-time relation of a specific feature point. The model is updated with new information. The system then predicts searching regions of the parts for the analysis of next image. If the prediction fails for a specific part, then the system terminates the analysis process for this part and works for others hoping that the information from succeeding images will resolve the confusion. The model presented here is essentially an attributed grammar, but with a slightly different form. Since it is focused on a specific object--human body, composed of 24 segments and 25 joints, the dynamic process of the image sequence is represented as a structure-preserved variation of attributed language. In other words, each image is represented by a set of feature values and the whole sequence is represented by a set of piecewise linear equations. Besides, it is not known whether or not the relaxation process of uncertain part will eventually be stable, which is the common problem in Yachida et al. [77] and Tsotsos et al. [74].

Tsotsos et al. [74] report a system for the analysis of the image sequence of left ventricular wall motion. They propose a way to represent motion concepts based on the semantic network theory. Each motion concept is associated with and defined by a "frame." A frame is definable by the user. Frames have an arbitrary number of "slots" that form their parts. Slots come in two varieties: "prerequisite" and "dependent." Prerequisite slots specify concepts that must be observed before the frame can be instantiated, while dependents provide additional semantic components that are included along with the frame concept on instantiation. For example, the concept of "area change" and "contract" is defined as follows:

```
frame area_change with
  prerequisites
       subj:  contractile_object;
       time_int:  time_interval;
       start_a:  area_value;
       end_a:  area_value;
end
frame contract is_a area_change with
   prerequisites
        start_a:  such that
                     start_a > end_a;
   dependents
        speed:  speed_v with
                   speed ← (start_a-end_a) ÷ time_int. duration
end
```

These two frames illustrate most of the "syntactical" construct of their representation formalism. From the viewpoint of Fu [10] and Tsai and Fu [14], the representation method of motion information described above is essentially that: each motion concept is represented as a set of prespecified attributes, each of which has a special meaning. Tsotsos et al. analyze the image sequence using hypotheses-cooperation method: uncertain parts in an image are updated using a relaxation process [88], during which the iteration is not completed until a convergence to stable condition is achieved. It is not sure whether or not the iteration process will always lead to convergence in every uncertainty case.

Yachida, Ikeda and Tsuji [76] analyze the behavior of heart wall motion by measuring the thickness of heart wall in each image. The thickness of the heart wall is measured at a set of boundary points. The thickness is determined as the distance between the points of the internal boundary and the points of the external boundary intersecting with a set of lines perpendicular to the internal boundary. After the feature extraction of the whole image sequence, the dynamic process of the heart wall motion is represented as a 3-D surface map, with x-axis being the spatial domain, y-axis being the time domain and z-axis being the thickness. The motion of heart wall is analyzed by observing the surface configuration. Heintzen et al. [93] also represent the information extracted from the heart motion images as a 3-D surface, while Garrison et al. [90,91,92,94,95] use a curve (volume-time) to represent the dynamic process.

The various research areas of time-varying images are briefly described as follows:

Aerial and satellite images: Evaluation of aerial and satellite image sequences is an active area in the problem of time-varying image analysis [15,18,20]. Wind velocities are estimated from cloud displacements observed in a sequence of satellite images as regular input data to weather forecasting. Hubert et al. [21] combine two or more copies from a sequence of satellite images taken at 20-30 minute intervals over a period of 2-3 hours into a film loop which is continuously projected onto a digitizing tablet. An operator selects a cloud feature (e.g. cyclonic vortices, cold fronts) and marks its position in the projection of initial and final frame from the film loop , using the intermediate frames to securely track the selected features. The measured displacement of the selected feature is corrected for distortion and transformed to an earth surface coordinate system. On the hypothesis that positional changes of clouds are solely affected by horizontal winds, the resulting cloud feature displacement is converted into wind velocity.

Traffic monitoring: Several investigations have been reported to detect moving vehicles in sequences of film or video-films of traffic scenes and to track their motion. A reliable solution to this problem would not only allow to count vehicles, but in addition enable the identification of vehicle type (car, truck,bus ,...) and the observation of vehicle behavior for a variety of quickly changing traffic situations. Wolferts [22] described an interactive setup to measure vehicle velocity on time-lapsed film using cross-correlation to track their images. Onoe et al. [23] evaluated video sequences for this purpose. Jones [24] reported on real-time tracking of features on vehicles in video-sequences. Jain et al. [25] used video-sequences of traffic scenes to study the detection, isolation, tracking, and description of moving objects.

Industrial automation: Application of visual sensors to industrial automation appears to be another area of time-varying image problems [26]. Newmann [27] studies the tracking of an object on a simulated conveyor belt by instantiating 2-D relational models on straight line contour approximations to images of a scene that contained a moving object. If tracking has to be performed under real-time constraints, one reverts to registering greyvalue or binary templates or to the identification of a feature that is characteristic for the object in question and can be easily isolated within a search window. Uno [28] employed such techniques on using a TV-camera to detect the position of bolts and reinforcement ribs on moving steel moulds in order to control a manipulator. Jones et al. [24] investigated real-time tracking techniques to study the application of fast visual feedback to industrial automation. Eskenazi [29] investigated a real-time tracking capability based on greyvalue correlation by using the video sensing and processing setup developed for the navigable robot. By simultaneously tracking the same object in two stereo image sequences, it is able to determine the object's 3-D trajectory. Using this technique to track a stationary object from a moving robot, the robot's trajectory can be determined in order to guide its navigation and to adjust the pan and tilt of a stereo camera assembly on the robot for keeping the reference object in the field of view.

Medical applications: Medicine provides a major area for image sequence analysis. The subtraction of X-ray images obtained before and after injection of roentgen-opaque material into the blood enhances the resulting difference image for interpretation by a physician. Digitization of X-ray film images facilitates the performance of nonlinear operations such as compensation for film characteristics during the determination of the difference image. Although digitized film images have to be aligned prior to subtraction, relative geometric distortions can usually be neglected. Therefore, the registration tends to be much less involved here than with satellite images. If an average of several digitized pre-injection images is subtracted from a post-injection image, image components not affected by the injection of roentgen-opaque material can be suppressed even more effectively [30,31].

Sequences of X-ray images from the left ventricle have been evaluated interactively by cardiologists to determine the left ventricle volume or to search for abnormalities in ventricle wall motion. One of the main problems consists in a reliable determination of the ventricle outline in the X-ray image. Chow and kaneko [30] search a suitably enhanced digitized X-ray film image of the left ventricle for subregions with larger greyvalue variances. For such subregions, a mixture of two normal distributions is fitted to the corresponding greyvalue histogram. If an acceptable fit is obtained for a subregion a threshold is determined which is used to classify the greyvalues of this subregion as either interior or exterior to the left ventricle. In this way, a series of 25 left ventricular contours is derived automatically covering approximately one cardiac cycle.

Injection of radioactive nuclei into a peripheral vein and subsequent recording of their decay gamma rays by a scientigraphic camera provides another way to obtain images of the left ventricle. Although these scientigraphic images offer even less contrast than good X-ray images, Hachimura et al. [32] were able to determine the left ventricular contour in such scientigraphic image sequences. By detailed analysis of the left ventricular contour obtained at the end-diastole and the end-systole they attempt to classify the observed left ventricule wall motion into a normal and an abnormal category. Based on a time series of nine scientigraphic images observed over 100 msec intervals after the end-diastole they estimate the left ventricular volume and plot it versus time.

A 3-D distribution of greyvalues reflecting the spatial configuration of organs in a living body can be determined by computerized tomograpny. A three layer approximation to such a description for the thorax of a living dog is given by Johnson et al. [33]. If position ,size and shape of an organ varies with time as in the case of the left ventricle, such a variation can be represented by a time series of 3-D greyvalue distributions which can be considered as a four-dimensional greyvalue distribution. For this purpose, Herman and Liu [34] generalized a search algorithm by Liu [35] for the determination of a 2-D surface in a 3-D greyvalue distribution such as obtained by 3-D computerized tomography.

Information about dynamic changes in size, shape and position on intact working organs is of great interest not only for the detection of abnormalities but even for a detailed understanding of their function. This is not restricted to time series of 3-D greyvalue distributions as obtained from computerized tomography. Heintzen et al. [36] emphasize this strong connection between time sequences of single-plane X-ray images and a better understanding of certain organ functions.

Beyond pure medical applications, the evaluation of image sequences spreads into biophysics and biology. For example, Yachida et al. [37] observe fishes swimming in a vat by an overhead TV-camera connected to a video-tape recorder to study their behavior under a variety of stimuli such as lights or tones. About 8 frames per second are recorded for periods between 2 and 30 seconds, resulting in sequences of 20-250 frames. Since the images of these moving objects are usually blurred, temporal as well as spatial greyvalue differences are used to separate the images of moving objects from those of stationary scene components. Results from a previous frame are used to guide the feature extraction process in the subsequent frame. They model the essential parts of the scene as it is presented in each frame and exploit such models obtained from neighboring - prior as well as posterior - frames to deduce uncertain parts or reanalyze them. Davenport et al. [38] also study the stimulus-response behavior of microorganisms whose movement in a wet-slide preparation under a microscope is recorded on video tape.

Ariki et al. [39] design an elaborate interactive facility for the analysis of image sequences. They analyze the morphogenetic movement of a dissociated cell of Xenopus laevis, a protozoon. Operator intervention is required to check frame registration (low level interaction). Medium level interaction is employed to define a model for the object that has to be traced throughout the image sequence. Such medium level interaction enable the evaluation of a variety of real world scenes. Takagi et al. [40] conduct a tracking problem in which to determine the paths of gramules in a cultivated, living pancreatic cell recorded by time-lapse cinemicrophotography at 1 second intervals for 5-10 minutes. These granules carry insulin from ribosomes where it is produced to the cell membrane. Analyzing the motion of these granules may therefore contribute to a better understanding of hormane production mechanism.

One of the problems in time-varying image analysis is the proper representation of information extracted from the image sequence. In syntactic pattern recognition, a pattern is usually represented by a linguistic notion called a sentence. The sentence could be a string, a tree or a graph of pattern primitives and relations. Most of the developments in syntactic pattern recognition research during the past two decades deal with 'static patterns' (e.g. fingerprint, character, chromosome shape). For static patterns, given an input pattern x and a set of pattern grammars representing different classes, parsing scheme is applied for the classification of input pattern. While for time-varying patterns, after finding a proper representation, we need to not only analyze the pattern at each single stage, but also test the mechanism characterizing the dynamic process of the sequence.

1.3 Summary of the Contents

Chapter 2 is concerned with the time-varying pattern analysis using a syntactic method. (For convenience, we use TV as an abbreviation of time-varying. In formal language theory, a translation is defined as a mapping from a language L_1 to another language L_2. If we consider L_1 as the set of possible patterns occurring at time t_1, L_2 as the set of patterns at time t_2, then the relation governing the TV phenomena could be formulated as a translation problem. In chapter 2, we first explore such a translation using a pattern deformation model [11] and then consider the applicability of the translation models to time-varying pattern analysis. Furthermore, in terms of the translation models, we will formulate the problem of TV pattern analysis as one which can be solved using traditional method, specifically, context-free programmed language parsing.

In chapter 3, the concept of language translation is extended from strings to trees using a generalized syntax-directed model. Conventional top-down and bottom-up tree translation models are compared with the generalized model. It is shown that both top-down and bottom-up models are special cases of the generalized model. A parsing algorithm for this generalized tree translation model is also presented. A traffic monitoring experiment is described in chapter 4. Each image of the traffic scene is divided into a set of windows which is then represented as a tree structure. A tree translation schema is applied to describe the motion of vehicles in the input image sequence. the matching process of vehicles between consecutive images is performed through a tree translation parsing.

Chapter 5 summarizes the results of this study and proposes suggestions for further research.

CHAPTER 2

SYNTAX OF TIME-VARYING PATTERNS

2.1 Introduction

In syntactic pattern recognition, a pattern is usually represented by a linguistic notion called a sentence [10]. The sentence could be a string, a tree, or a graph of pattern primitives and relations. Most of the developments in syntactic pattern recognition research during the past two decades deal with "static patterns" (e.g. chromosome classification, character recognition, and fingerprint classification). For static patterns, given an input pattern x and a set of pattern grammars representing different classes, error-correcting parsers are used to classify x into one of the classes [10,11]. While for TV patterns, given an input sequence x_1, x_2, ..., we need to not only analyze the pattern at each single stage, but also test the mechanism characterizing the dynamic process of the sequence.

In formal language theory, a translation is defined as a mapping from a language L_1 to another language L_2. If we consider L_1 as the set of possible patterns occurring at time t_1, L_2 as the set of patterns at time t_2, then the relation governing the TV phenomena could be formulated as a translation problem. In this chapter, we first explore such a translation using a pattern deformation model [11] and then consider the applicability of the translation models to time-varying pattern analysis. Furthermore, in terms of the translation models, we will formulate the problem of TV pattern analysis as one which can be solved using traditional method, specifically, context-free programmed language parsing.

2.2 Preliminaries

Definitions and notations that will be referred to in this chapter are summarized as follows [51-54].

Definition 2.1: Suppose that Σ is an input alphabet and Δ is an output alphabet. We define a translation from a language $L_1 \subseteq \Sigma^*$ to a language $L_2 \subseteq \Delta^*$ as a relation T from Σ^* to Δ^* such that the domain of T is L_1 and the range of T is L_2.

Definition 2.2: A syntax-directed translation schema (SDTS for short) is a 5-tuple $T = (N, \Sigma, \Delta, R, S)$, where

(1) N is a finite set of nonterminal symbols,

(2) Σ is a finite input alphabet,

(3) Δ is a finite output alphabet,

(4) R is a finite set of rules of the form $A \rightarrow \alpha, \beta$, where $\alpha \in (N \cup \Sigma)^*$, $\beta \in (N \cup \Delta)^*$, and the nonterminals in β are a permutation of the nonterminals in α,

(5) S is a distinguished nonterminal in N, the start symbol.

Let $A \rightarrow \alpha, \beta$ be a rule. To each nonterminal of α there is an associated identical nonterminal of β. If a nonterminal B appears only once in α and β, then the association is obvious. If B appears more than once, we use integer superscripts to indicate the association. This association is an intimate part of the rule. For example, in the rule $A \rightarrow B^{(1)}CB^{(2)},\ B^{(2)}B^{(1)}C$, the three positions in $B^{(1)}CB^{(2)}$ are associated with positions 2, 3, and 1, respectively, in $B^{(2)}B^{(1)}C$. The translation defined by T, denoted $\tau(T)$, is the set of pairs

$$\{(x,y) \mid (S,S) \overset{*}{\Longrightarrow} (x,y),\ x \in \Sigma^{*} \text{ and } y \in \Delta^{*}\}.$$

Definition 2.3: If $T = (N, \Sigma, \Delta, R, S)$ is an SDTS, then $\tau(T)$ is called a syntax-directed translation (SDT). The grammar $G_i = (N, \Sigma, P, S)$, where

$$P = \{A \rightarrow \alpha \mid A \rightarrow \alpha, \beta \text{ is in } R\},$$

is called the input grammar of the SDTS T. The grammar $G = (N, \Delta, P', S)$, where $P' = \{A \rightarrow \beta \mid A \rightarrow \alpha, \beta \text{ is in } R\}$ is called the output grammar of T.

Definition 2.4: An SDTS $T = (N, \Sigma, \Delta, R, S)$ such that in each rule $A \rightarrow \alpha,\ \beta$ in R, associated nonterminals occur in the same order in α and β is called a simple SDTS. The translation defined by a simple SDTS is called a simple SDT.

Definition 2.5: A simple SDTS $T = (N, \Sigma, \Delta, R, S)$ such that each rule in R is either of the form $A \rightarrow aB, \alpha B$ or of the form $A \rightarrow a, \alpha$ where $A, B \in N$, $a \in \Sigma$ and $\alpha \in \Delta^{*}$, is called a regular SDTS. The translation defined by a regular SDTS is called a regular SDT.

Definition 2.6: For two strings, x, y ϵ Σ^*, we define a transformation T: $\Sigma^* \rightarrow \Sigma^*$ such that y ϵ T(x). The following three transformations are introduced [54]:

(1) substitution error transformation

$$\omega_1 a \omega_2 \overset{T_S}{\vdash\!\!-\!-\!-} \omega_1 b \omega_2, \text{ for all } a, b \in \Sigma, a \neq b,$$

(2) deletion error transformation

$$\omega_1 a \omega_2 \overset{T_D}{\vdash\!\!-\!-\!-} \omega_1 \omega_2, \text{ for all } a \in \Sigma \quad ,$$

(3) insertion error transformation

$$\omega_1 \omega_2 \overset{T_I}{\vdash\!\!-\!-\!-} \omega_1 a \omega_2, \text{ for all } a \in \Sigma \text{ where } \omega_1, \omega_2 \in \Sigma^*.$$

Example 2.1: Given a sentence x = cbabdbb and a sentence y = cbbabbdb, then

$$x = \text{cbabdbb} \overset{T_S}{\vdash\!\!-\!-\!-} \text{cbabbbb} \overset{T_S}{\vdash\!\!-\!-\!-} \text{cbabbdb} \overset{T_I}{\vdash\!\!-\!-\!-} \text{cbbabbdb} = y \quad .$$

Definition 2.7: A context-free programmed grammar (CFPG) is a 5-tuple G = (N, Σ, J, P, S) where

(1) N is a finite set of non-terminals,

(2) Σ is a finite set of terminals,

(3) S is the start symbol in N,

(4) P is a finite set of programmed productions,

(5) J is a finite set of production labels.

Each production in P consists of a label $r \epsilon J$, a core production of the form $A \rightarrow \alpha$ where $A \epsilon N$, $\alpha \epsilon (N \cup \Sigma)^*$, and a success branch field and a failure branch field each consisting of elements from J. (This definition allows core production to be $A \rightarrow \lambda$, where $A \epsilon N$ and λ is an empty string.)

A derivation or generation in G proceeds as follows: The first production is applied to the start symbol S; therefore, if production r is applied to the current sentential form γ to rewrite a nonterminal A, and if γ contains at least one occurrence of A, then the leftmost A is rewritten by the core of production r and the next production label is selected from the success branch field of r; if the current sentential form does not contain A, then the core of production r cannot be used and the next production label is selected from the failure branch field of r; if the applicable branch field is empty, the derivation halts.

2.3 Formulation of TV Patterns as a CFPL

To recognize noisy syntactic patterns, various types of error transformation models were proposed. Fung and Fu [52] proposed a structure-preserved deformation model to handle substitution error for string patterns. Lu and Fu [11] extended the model to include substitution error, insertion error and deletion error, which were introduced in the previous section. It is known that any pattern x can be transformed into any other pattern y by a sequence of error transformations. If we analyze a TV pattern $x_1, x_2, \ldots x_n$ from the viewpoint of deformation model, then the TV phenomena can well be interpreted through a sequence of properly selected error transformations which transforms x_1 into x_2, x_2 into x_3, ..., and x_{n-1} into x_n.

The deformation models proposed by Fung and Fu, and Lu and Fu can be formulated in terms of translation schema. Consider patterns which are described by a regular grammar. To handle substitution error, a regular SDTS $T(N, \Sigma, \Delta, R, S)$ can be constructed, where $\Sigma = \Delta$ is the primitive set, R is a set of rules of the form

$$A \to aB,\ bB \qquad A, B \in N,\ a, b \in \Sigma$$

For each rule $A \to aB$ in the regular grammar and each terminal $b \in \Sigma$, add a rule $A \to aB$, bB to R. To include insertion error and deletion error, for each rule $A \to aB$ in the regular grammar, add a rule $A \to aB$, αB to R for each α in $(\Sigma^1 \cup \Sigma^2 \cdots \cup \Sigma^K \cup \{e\})$, where K is a parameter and e is the empty string. For patterns which are described by a context-free grammar, a simple SDTS can be constructed to model the three types of deformation. First of all, the grammar is transformed into its Greibach Normal Form [51]. Then the construction procedure of a simple SDTS is similar to that of a regular translation schema except that: For each rule $A \to a\beta$ in the context-free grammar, add $A \to a\beta$, $\alpha\beta$ for each $\alpha \in (\Sigma^1 \cup \Sigma^2 \cdots \cup \Sigma^K \cup \{e\})$.

As far as error modeling is concerned, Lu and Fu's model can handle any number of errors occurred. For time-varying patterns, we restrict the number of errors no larger than K. The difference between Lu and Fu's model and this translation model is that Lu and Fu's model handle errors implicitly (or recursively) while this translation model must explicitly specify the kind of errors it intends to cover. For example, in Lu and Fu's model $E_a \to a$ and $E_a \to bE_a$ recursively describe any number of errors of inserting b in front of a. But in this translation model, a translation rule $A \to a\beta, \alpha\beta$ has to be included to cover each individual error $a \to \alpha$ even though α has no limitation. The parameter k here only indicates that the number of errors $a \to \alpha$ is finite. In other words, k could be any positive integer but $k < \infty$. Essentially we use the translation schema to implement the deformation model. To make it suitable for an n-stage TV pattern analysis problem with sequence x_1, x_2, ..., x_{n+1}, we give the following definition.

Definition 2.8: An n-stage SDTS is a 5-tuple

$$T = (N, \Sigma, \Delta, R, S), \text{ where}$$

(1) N, Σ, Δ, S are defined as those of an SDTS.

(2) R is a set of rules of the form $A \to \alpha, \beta_1, \beta_2, \ldots, \beta_n$. where $\alpha \in (N \cup \Sigma)^*$, $\beta_i \in (N \cup \Delta)^*$, and the nonterminals in β_i are a permutation of the nonterminals in α.

The translation defined by an n-stage SDTS T, denoted by $\tau(T)$, is the set

$$\{(x, y_1, \cdots y_n) \mid (S, S, \cdots, S) \overset{*}{\Longrightarrow} (x, y_1, \cdots, y_n), x \in \Sigma^*, y_i \in \Delta^*\}$$

Thereafter the analysis of TV patterns becomes a translation problem.

Next we investigate the "parsing" of a translation beginning with a definition:

Definition 2.9: Let T be a one-stage SDTS

$$\tau(T) = \{(x, y) \mid (S, S) \overset{*}{\Longrightarrow} (x, y)\}.$$

Define the input language

L_1 as $\{x \mid (x, y) \in \tau(T)$ for all $y\}$

and the output language

L_2 as $\{y \mid (x,y) \in \tau(T)$ for all $x\}$.

The concatenation of L_1 and L_2 with respect to T is defined as the set $\{xy \mid (x, y) \in \tau(T)\}$, denoted by L_{12}. $L_{12} \subseteq L_1 \cdot L_2$, where $L_1 \cdot L_2 = \{xy \mid x \in L_1, y \in L_2\}$.

Theorem 2.1: Let L_1 and L_2 be the input language and output language respectively of a regular SDTS $T(N, \Sigma, \Delta, R, S)$ then L_{12} is a CFPL. (L_1 is a context-free language.)

Proof: We shall prove the theorem by constructing a CFPG G such that $L(G) = L_{12}$.

A CFPG $G(N', \Sigma', J, P, S)$ is constructed as follows:

Step 1: $N' = \{A^1 \mid A \in N\} \cup \{A^2 \mid A \in N\} \cup \{S\}$.

Step 2: $\Sigma' = \Sigma \cup \Delta$.

Step 3: Since T is a regular SDTS, each production in R is of the form:

$A \rightarrow aB, \alpha B$ or $A \rightarrow a, \alpha$ where $A, B \in N$, $a \in \Sigma$, $\alpha \in \Delta^*$.

(1) Add $S \rightarrow S^1S^2$ $\{K|K$ is the label of production beginning with $S^1\}$ ϕ to P.

(2) If $A \rightarrow aB, \alpha B$ is a production in R such that $A, B \in N$, $a \in \Sigma$ and $\alpha \in \Delta^*$, then add the production to P:

$A^2 \rightarrow \alpha B^2$ $\{K|K$ is the label of production beginning with $B^1\}$ ϕ

(3) For each input rule of R: $A \rightarrow aB$ where $A, B \in N$ and $a \in \Sigma$, add the production to P: $A^1 \rightarrow aB^1$ $S(u)$ ϕ, where $S(u)$ is the set of labels of the productions added in (2) corresponding to the translation rules with input rule $A \rightarrow aB$.

(4) If $A \rightarrow a, \alpha$ is a production in R, where $A \in N$, $a \in \Sigma$ and $\alpha \in \Delta^*$, then add the production $A^2 \rightarrow \alpha$ ϕ ϕ to P.

(5) For each input rule of R: $A \rightarrow a$ where $A \in N$ and $a \in \Sigma$, add the production to P: $A^1 \rightarrow a$ $S(u)$ ϕ where $S(u)$ is the set of labels of the productions added in (4) corresponding to the translation rules with input rule $A \rightarrow a$.

Then we shall prove that $L_{12} = L(G)$.

Let xy be in L_{12}. Then by Definition 2.9 there exists a derivation sequence

$$(S, S) \Longrightarrow (a_1A_1, \alpha_1A_1) {---} \Longrightarrow (a_1a_2{---}a_n, \alpha_1\alpha_2{---}\alpha_n) = (x,y).$$

where $a_i \in \Sigma$, $\alpha_i \in \Delta^*$ and $A_i \in N$. From the construction procedure of CFPG G, it is known that: For each derivation step

$$(a_1a_2{---}a_iA_i, \alpha_1\alpha_2{---}\alpha_iA_i) \Longrightarrow (a_1a_2{---}a_{i+1}A_{i+1}, \alpha_1\alpha_2{---}\alpha_{i+1}A_{i+1}),$$

there exist the productions

$$A_i^1 \to a_{i+1} A_{i+1}^1 \quad \text{and} \quad A_i^2 \to \alpha_{i+1} A_{i+1}^2 \text{ in } P$$

with the success field of the first production containing the label of the second production. Besides the success field of the production $A_i^2 \to \alpha_{i+1} A_{i+1}^2$ contains the label of the production $A_{i+1}^1 \to a_{i+2} A_{i+2}^1$ for $1 \leq i < n$. Therefore, there exists a derivation in G: $S \Longrightarrow S^1S^2 \Longrightarrow a_1A_1S^2 \Longrightarrow a_1A_1^1\alpha_1A_1^2 \Longrightarrow a_1a_2A_2^1\alpha_1A_1^2 \Longrightarrow a_1a_2A_2^1\alpha_1\alpha_2A_2^2$ $\Longrightarrow \cdots\cdot$ $\Longrightarrow a_1a_2 \cdots a_n\alpha_1\alpha_2 \cdots \alpha_{n-1} A_{n-1}^2$ $\Longrightarrow a_1a_2 \cdots a_n\alpha_1\alpha_2 \cdots \alpha_{n-1}\alpha_n = xy$. Thus, $L_{12} \subseteq L(G)$. Similarly, from the definition of G, it can be shown that $L(G) \subseteq L_{12}$. Therefore, $L_{12} = L(G)$.

Definition 2.10: Let T be an n-stage SDTS, L_1 be its input language, L_2, $L_3 \cdots L_{n+1}$ be its 1st, 2nd, ⋯ nth output languages respectively. Define the concatenation of $L_1, L_2, \cdots L_{n+1}$ with respect to T as $\{x_1x_2 \cdots x_{n+1} \mid (x_1, x_2, \cdots, x_{n+1}) \in \tau(T)\}$ denoted as $L_{123\cdots n+1}$.

Theorem 2.2: Let T be an n-stage regular SDTS, $L_1, L_2 \cdots, L_{n+1}$ be its input language, 1st, ⋯ nth output languages respectively. Then $L_{12\cdots n+1}$ is a CFPL. The proof is similar to that of Theorem 2.1 except that for the present case, $N' = \bigcup_{i=1}^{n} \{A^i \mid A \in N\} \cup \{S\}$, and for each production in T, there are n+1 corresponding productions in G. (L_1 is a context-free language.)

Theorem 2.3: Let T be a simple SDTS, L_1, L_2 be its input and output languages respectively. Then L_{12} is a CFPL. (L_1 is a context-free language.)

Proof: We shall construct a CFPG G such that $L(G)=L_{12}$. Since T is a simple SDTS, each production in T is of the form $A \to \alpha, \beta$ where $A \in N$, $\alpha \in (N \cup \Sigma)^*$, $\beta \in (N \cup \Delta)^*$, α, β have the same nonterminals with the same permutation. $G(N',\Sigma',J,P,S)$ is constructed as follows:

Step 1. $N' = \{A^1 | A \in N\} \cup \{A^2 | A \in N\} \cup \{S\}$.

Step 2. $\Sigma' = \Sigma \cup \Delta$.

Step 3. (1) Add $S \to S^1S^2$ $\{K|K$ is the label of production beginning with $S^1\}$ ϕ to P.

(2) For each production $A \to \alpha_1A_1 \ldots \alpha_nA_n\alpha_{n+1}, \beta_1A_1 \ldots \beta_nA_n\beta_{n+1}$ in R, where $A_i \in N$, $\alpha_i \in \Sigma^*$, $\beta_i \in \Delta^*$, $A \in N$, add

$A^1 \to \alpha_1A_1^1 \ldots \alpha_nA_n^1\alpha_{n+1}$ $\{K|K$ is the label of production $A^2 \to \beta_1A_1^2 \ldots A_n^2\beta_{n+1}\}$ ϕ

and $A^2 \to \beta_1A_1^2 \ldots \beta_nA_n^2\beta_{n+1}$ $\{K|K$ is the label of productions beginning with the leftmost nonterminal of the present derivation string$\}$ ϕ

Strictly speaking, there is little restriction on the next step of a derivation after one application of the production beginning with a nonterminal of the form A^2 where A is in N. The only restriction is that the next applied production must begin with a nonterminal of the form A^1 where A is in N.

The proof that $L(G)=L_{12}$ is similar to that of Theorem 2.1.

Corollary 2.1: Let T be an n-stage simple SDTS, L_1 be its input language, and L_2, ... L_{n+1} be its 1st, 2nd ... nth output languages respectively. Then $L_{12...n+1}$ is a CFPL.

Rosenkrantz [53] proposed the concept of programmed grammars as a new device for generating languages. It is shown that the class of CFPL properly contains the class of context-free languages and is properly contained within the class of context-sensitive languages. Writing a CFPG is very much like writing a computer program and is a rather straightforward logical process compared with the task of writing a context-sensitive grammar. Swain and Fu [54] further applied CFPG to pattern recognition and proposed a stochastic syntax analyzer for CFPL. Lu and Fu [11] then proposed an error-correcting parsing algorithms for CFPL. By the previous discussions, it is known that a class of TV patterns can be described as a CFPL. Then the analysis of TV patterns becomes the parsing of a CFPL which has been well developed by Fu et al. This fact is to say that some TV pattern analysis problems can be thought of as parsing of a CFPL. In other words, we can infer a CFPG to interpret or to generate the time-varying process. A CFPL parser can be used to classify an input pattern sequence. Actually the parsing task can be performed by directly building a parsing algorithm from the translation schema instead of constructing a CFPG. This parser consists of n interrelated "sub-parser" each of which deals with the corresponding x_i input string. The basic idea is that these n input strings $x_1, ..., x_n$ need not be concatenated. However the relation among these subparsers still has the spirit of a CFPL parser.

In applying syntactic methods to pattern recognition, one dimensional (string) grammars are sometimes inefficient in describing high-dimensional patterns. For the purpose of effectively describing high-dimensional patterns, Fu and Bhargava [55] introduced the application of tree systems to pattern recognition. Lu and Fu [12] proposed error-correcting tree automata for the recognition of noisy patterns. Five types of error transformation are introduced:

(1) the substitution of the label of a node by another terminal symbol,

(2) the insertion of an extraneous labeled node between a node and its immediate predecessor,

(3) the insertion of an extraneous labeled node to the left of all the immediate successors of a node,

(4) the insertion of an extraneous labeled node to the right of a node,

(5) the deletion of a node of rank 1 or 0.

The three operations of insertions in rule (2), (3) and (4) are named as stretch, branch, and split, respectively, according to the relative position of the inserted node to the original tree. Apparently, the inverse operation of any type of insertion is deletion, and the inverse of deletion operation is one of the three types of insertion.

The deformation from a pure pattern to its noisy pattern can be performed through a sequence of application of these error transformations. Through the use of the deformation model, the transformation concept can also be applied to tree grammar to solve some time-varying pattern analysis problems which are not easy to handle with string grammar translations. After introducing a brief definition, we propose a tree grammar translation schema. (For detailed description about tree grammar, refer to [12]).

Definition 2.11: A tree grammar $G_t = (V,r,P,S)$ over $<\sum,r>$ is expansive iff each production in P is of the form

$$X_0 \rightarrow \begin{array}{c} x \\ / \ \backslash \\ X_1 \ldots X_{r(x)} \end{array} \qquad \text{or } X_0 \rightarrow x \text{ where } x \epsilon \sum$$

and $X_0, X_1, \ldots, X_{r(x)}$ are nonterminal symbols.

Definition 2.12: A tree SDTS T is a 4-tuple (V,r,R,S) where V,r,S have the same meaning as for a tree grammar, R is a set of productions of the form: $A \rightarrow \alpha, \beta$ such that α and β have the same nonterminals, β is the result of application of a sequence of error transformations to α and the input grammar is in the expansive form.

An n-stage tree SDTS can be formed with each production being of the form $A \rightarrow \alpha, \beta_1, \beta_2 \ldots \beta_n$ where each β_i is the result of some deformation of α. The concept of "programmedness" can also be applied to tree grammar. Consequently, Lu and Fu's tree language analysis method can be used to analyze some more complex TV patterns.

2.4 Stochastic Translation

Definition 2.13: A stochastic syntax-directed translation schema (SSDTS) [83] is a 5-tuple $T = (N,\sum,\Delta,R,S)$ where

N = a finite set of nonterminals;

$\sum$ = a finite input alphabet;

Δ = a finite output alphabet;

R = a finite set of rules of the form $p: A \rightarrow \alpha,\beta$ for A in N, α in $(N \cup \Sigma)^*$, β in $(N \cup \Delta)^*$, $0 < p \leq 1$, with the nonterminals in α being a permutation of those in β;

S in N = the starting symbol.

Each nonterminal in the α portion of a rule has an identical, associated one in β (matching superscripts are used as needed for repeated nonterminals).

Definition 2.14: A schema T is:

(a) unrestricted if for each rule $p: A \rightarrow \alpha,\beta$ the probability p is not conditioned on other rules or events.

(b) proper if for each nonterminal A the probabilities of all rules in which A is the leftside nonterminal sum to 1.

We consider only unrestricted, proper schema. Translations from Σ^* to Δ^* are produced as follows.

Definition 2.15: A translation form of T is defined recursively:

(a) 1: (S,S) is a form with associated S's;

(b) if $p: (\omega A \sigma, \gamma A \delta)$ is a form with the A's associated and $\hat{p}: A \rightarrow \alpha,\beta$ is a rule, then $p\hat{p}: (\omega\alpha\sigma, \gamma\beta\delta)$ is a form.

Definition 2.16: The stochastic translation from Σ^* to Δ^* produced by T with starting symbol S is the set

$$t(T,S) = \{(x,y,p) | x \text{ in } \Sigma^*, y \text{ in } \Delta^*, p = \Sigma \hat{p} \text{ for all translation forms } \hat{p}: (x,y)\}$$

T defines a function $p_T: \Sigma^* \times \Delta^* \rightarrow [0,1]$ such that

$$p_T(x,y) = \sum_{i=1}^{n} \prod_{j=1}^{f(i)} p_{ij}(x,y)$$

where there are n distinct standard derivations (e.g., leftmost) of the translation form (x,y) with f(i) steps in the ith one, and $p_{ij}(x,y)$ is the probability assigned to the jth rule of the ith derivation.

Definition 2.17: A schema T is:

(a) consistent if

$$\sum_x \sum_y p_T(x,y) = 1.$$

(b) simple if in each rule p: $A \to \alpha,\beta$ the nonterminals in α occur in the same order as their associates in β.

(c) regular if all rules have the form p: $A \to xB,\ yB$ or p: $A \to x,y$ for A, B in N, X in Σ, y in Δ^*.

Example 2.2: The occurrence of substitution error in a 4-bit binary code transmission can be described as follows:

$$T = (\{S,A,B,C\},\ \{0,1\},\ \{0,1\},\ R,S)$$

R : $S \xrightarrow{\frac{1}{2}P_c} 0\,A,\ 0\,A$

$S \xrightarrow{\frac{1}{2}P_s} 0\,A,\ 1\,A$

$S \xrightarrow{\frac{1}{2}P_c} 1\,A,\ 1\,A$

$S \xrightarrow{\frac{1}{2}P_s} 1\,A,\ 0\,A$

$A \xrightarrow{\frac{1}{2}P_c} 0\,B,\ 0\,B$

$A \xrightarrow{\frac{1}{2}P_s} 0\,B,\ 1\,B$

$A \xrightarrow{\frac{1}{2}P_c} 1\,B,\ 1\,B$

$$A \xrightarrow{\frac{1}{2} P_s} 1\,B,\ 0\,B$$
$$B \xrightarrow{\frac{1}{2} P_c} 0\,C,\ 0\,C$$
$$B \xrightarrow{\frac{1}{2} P_s} 0\,C,\ 1\,C$$
$$B \xrightarrow{\frac{1}{2} P_c} 1\,C,\ 1\,C$$
$$B \xrightarrow{\frac{1}{2} P_s} 1\,C,\ 0\,C$$
$$C \xrightarrow{\frac{1}{2} P_c} 0,\ 0$$
$$C \xrightarrow{\frac{1}{2} P_s} 0,\ 1$$
$$C \xrightarrow{\frac{1}{2} P_c} 1,\ 1$$
$$C \xrightarrow{\frac{1}{2} P_s} 1,\ 0$$

where P_c is the probability of correct transmission of each individual bit, P_s is the probability of substitution error for each individual bit and $P_c + P_s = 1$. The translation pair (0000, 0010) is derived as follows:

$$S \xrightarrow{\frac{1}{2} P_c} 0\,A,\ 0\,A$$
$$\xrightarrow{\frac{1}{2} P_c} 00B,\ 00B$$
$$\xrightarrow{\frac{1}{2} P_s} 000C,\ 001C$$
$$\xrightarrow{\frac{1}{2} P_c} 0000,\ 0010$$

The probability of this derivation is $\frac{1}{2} P_c \times \frac{1}{2} P_c \times \frac{1}{2} P_s \times \frac{1}{2} P_c = \frac{1}{16} P_c^3 P_s$.

There is another way to describe the probabilistic property of a stochastic SDTS. Suppose that A is a nonterminal of a stochastic SDTS and there are n translation rules with $A \rightarrow \alpha$ as leftside (or input rule as defined before):

$$A \xrightarrow{P_1} \alpha,\ \beta_1$$
$$A \xrightarrow{P_2} \alpha,\ \beta_2$$

$$\vdots$$

$$A \xrightarrow{P_n} \alpha, \beta_n$$

Then rewrite these rules as

$$A \xrightarrow{P} \alpha \;, \; A \xrightarrow{P_1/P} \beta_1$$
$$A \xrightarrow{P_2/P} \beta_2$$
$$\vdots$$
$$A \xrightarrow{P_n/P} \beta_n$$

where $P = \sum_{i=1}^{n} P_i$

This set of rules states that rule $A \to \alpha$ can be applied for rewriting of A in input sentence with probability p and if rule $A \to \alpha$ is applied, there are n alternative rules for rewriting of A in output sentence with $\frac{P_i}{p}$ being the conditional probability of the ith rule. This variation of rule format does not change the probability of the derivation of a translation pair. Suppose that $A \xrightarrow{P_i} \alpha, \beta_i$ is applied, then $A \xrightarrow{P} \alpha$ and $A \xrightarrow{P_i/P} \beta_i$ are also applied and $P_i = P \times \frac{P_i}{p}$.

One advantage of this new form of translation rule is that it can display the probability of translation pair, input sentence and the conditional probability of output sentence. For instance, the stochastic SDTS of example 2.2 can be rewritten as:

$$S \xrightarrow{\frac{1}{2}} 0\,A \;, \; S \xrightarrow{P_c} 0\,A$$
$$S \xrightarrow{P_s} 1\,A$$
$$S \xrightarrow{\frac{1}{2}} 1\,A \;, \; S \xrightarrow{P_c} 1\,A$$
$$S \xrightarrow{P_s} 0\,A$$
$$A \xrightarrow{\frac{1}{2}} 0\,B \;, \; A \xrightarrow{P_c} 0\,B$$

$$\begin{array}{lcl}
 & & A \xrightarrow{P_s} 1\,B \\
A \xrightarrow{\frac{1}{2}} 1\,B & , & A \xrightarrow{P_c} 1\,B \\
 & & A \xrightarrow{P_s} 0\,B \\
B \xrightarrow{\frac{1}{2}} 0\,C & , & B \xrightarrow{P_c} 0\,C \\
 & & B \xrightarrow{P_s} 1\,C \\
B \xrightarrow{\frac{1}{2}} 1\,C & , & B \xrightarrow{P_c} 1\,C \\
 & & B \xrightarrow{P_s} 0\,C \\
C \xrightarrow{\frac{1}{2}} 0 & , & C \xrightarrow{P_c} 0 \\
 & & C \xrightarrow{P_s} 1 \\
C \xrightarrow{\frac{1}{2}} 1 & , & C \xrightarrow{P_c} 1 \\
 & & C \xrightarrow{P_s} 0
\end{array}$$

The left part above is the set of rules for input language and each rule is followed by a set of rules for output language. The input string 0000 is derived as follows: $S \xrightarrow{\frac{1}{2}} 0\,A \xrightarrow{\frac{1}{2}} 00B \xrightarrow{\frac{1}{2}} 000C \xrightarrow{\frac{1}{2}} 0000$. The probability of this derivation is $\frac{1}{16}$. Given this derivation, the output string 0010 is derived as follows $S \xrightarrow{P_c} 0\,A \xrightarrow{P_c} 00B \xrightarrow{P_s} 001C \xrightarrow{P_c} 0010$. The conditional probability of the output string 0010 given that 0000 is the input string is $P_c^3 P_s$. The probability of the translation pair <0000,0010> is $\frac{1}{16}\, P_c^3 P_s$.

Definition 2.18: A stochastic programmed grammar [10] is a 5-tuple $G = (V_N, V_T, J, P, S)$, where

(1) V_N is a finite set of nonterminals;

(2) V_T is a finite set of terminals;

(3) J is a finite set of production labels;

(4) P is a finite set of productions;

(5) S is the start symbol, $S \in V_N$.

Each production in P is of the form

$$(r)\ \omega \rightarrow \eta \quad S(U)P(U) \quad F(W)P(W).$$

The core of the production is "$\omega \rightarrow \eta$," where

$$\omega \in (V_T \cup V_N)^* V_N (V_T \cup V_N)^*$$

and

$$\eta \in (V_T \cup V_N)^*.$$

Each stochastic production in P has a distinct label, $r \in J$. $U \in 2^J$ and $W \in 2^J$ are called the success and the failure "go-to" fields, respectively. P(U) and P(W) are probability distributions associated with the sets U and W, respectively.

In applying the stochastic production to an intermediate string ξ derived from S under leftmost interpretation, if ξ contains the substring ω, then the leftmost occurrence of ω is expanded into η and the next production to be applied is selected from the success "go-to" field U according to a probability distribution P(U). If ξ does not contain ω, then ξ is not changed and the next production is selected from the failure "go-to" field with a probability distribution P(W).

Suppose that $S = \omega_1 \xrightarrow{r_1} \omega_2 \xrightarrow{r_2} \cdots \xrightarrow{r_n} \omega_{n+1} = x$ is a derivation of x, where r_j denotes the label of the production used to directly derive ω_{j+1} from ω_j. The probability, $p_i(x)$, associated with the i-th derivation of x is defined as the product of the conditional probabilities $p(r_1)p(r_2|r_1)\ldots p(r_n|r_{n-1})$. The interpretation of the conditional probability $p(r_j|r_{j-1})$ is the probability of selecting r_j as the next production to be applied from the success "go-to" field if the r_{j-1}-th production is successfully applied. Otherwise, $p(r_j|r_{j-1})$ denotes the probability of selecting r_j as the next production to be applied from the failure "go-to" field of the r_{j-1}-th production. $p(r_1)$ is assumed to be 1.

Definition 2.19: A stochastic programmed grammar G is said to be a stochastic context-sensitive, context-free, or regular programmed grammar, if G is a context-sensitive, context-free or regular pg.

The stochastic language generated by a stochastic pg G is

$$L(G) = \{(x,p(x))|x \in V_T^*, S \xrightarrow{p_i(x)} x, \text{ for } i= 1,\ldots,k$$

and

$$\sum_{i=1}^{k} p_i(x) = p(x)\},$$

and k is the number of all distinctly different derivations of x from S defined in G and $p_i(x)$ is the probability associated with the i-th derivation of x.

The stochastic language L(G) is called a stochastic programmed language. G is said to be consistent if and only if

$$\sum_{x \in L(G)} p(x) = 1.$$

Theorem 2.4: Let L_1 and L_2 be the input language and the output language, respectively, of a stochastic regular translation schema $T(N,\Sigma,\Delta,R,S)$, then L_{12} is a stochastic context-free programmed language (SCFPL). (L_1 is a context-free language.)

Proof: We shall prove the theorem by constructing a SCFPG G such that $L(G) = L_{12}$. A SCFPG $G(N',\Sigma',J,P,S)$ is constructed as follows:

Step 1. $N' = \{A^1 | A \in N\} \cup \{A^2 | A \in N\} \cup \{S\}$.

Step 2. $\Sigma' = \Sigma \cup \Delta$.

Step 3. Since T is a stochastic regular translation schema, each production in R is of the form

$$P: \quad A \to aB,\ \alpha B$$

or

$$P: \quad A \to a,\ \alpha \text{ where } A,B \in N,\ a \in \Sigma,\ \alpha \in \Delta^*.\ 0 \le P \le 1\ .$$

(1) Put all rules of R which have the same leftside nonterminal in a group within which all rules with the same input rule are further grouped together as follows:

P_{11}: $A \to a_1B_1,\ \alpha_{11}B_1$

P_{12}: $A \to a_1B_1,\ \alpha_{12}B_1$

⋮

$P_{1n(1)}$: $A \to a_1B_1,\ \alpha_{1n(1)}B_1$

P_{21}: $A \to a_2B_2,\ \alpha_{21}B_2$

⋮

$P_{mn(m)}$: $A \to a_mB_m,\ \alpha_{mn(m)}B_m$

or

$P_{mn(m)}$: $A \rightarrow a_m, \alpha_{mn(m)}$

where $0 \leq P_{ij} \leq 1$,

$A, B_i \in N$,

$a_i \in \Sigma$,

$\alpha_{ij} \in \Delta^*$,

$n(i)$ is the number of rules having the

same input rule $A \rightarrow a_iB_i$ or

$A \rightarrow a_i$,

and m is the number of different input rules

for nonterminal A,

$1 \leq i \leq m$, $1 \leq j \leq n(i)$.

(2) For each rule P_{ij}: $A \rightarrow a_iB_i, \alpha_{ij}B_i$ of the group formed for A in (1), add the rule to P:

$$\gamma \quad A^2 \rightarrow \alpha_{ij} \; B_i^2 \quad S(U) \quad P(U) \quad \phi \quad 1$$

where γ is the assigned label for this new rule, $S(U)$ and $P(U)$ are the success field and the associated probability distributions. The construction of $S(U)$ and $P(U)$ is explained by the following example: Suppose that $B_i = A$ and the set of rules in R is grouped as shown in (1). Then $S(U) = \{\gamma(1),\gamma(2),\ldots,\gamma(m)\}$ and $P(U) = \{P_1,P_2,\ldots,P_m\}$ where $\gamma(k)$ is the label for the corresponding rule of the k-th input rule of A, which will be described in (3), and $P_i = P_{i1} + P_{i2} + \ldots + P_{in(i)}$ is the corresponding probability.

(3) For each different input rule $A \rightarrow a_iB_i$ of the group formed for A in (1), add the rule to P:

$$\gamma \quad A^1 \rightarrow a_i B_i^1 \quad S(U) \quad P(U) \quad \phi \; 1$$

where γ is the assigned label for the rule, S(U) and P(U) are the success field and the associated probability distribution respectively. $S(U) = \{\gamma(1),\gamma(2),\ldots,\gamma(n(i))\}$, where n(i) is the number of translation rules with the same input rule $A \rightarrow a_i B_i$ and $\gamma(k)$ is the label of the rule in P corresponding to the k-th of these translation rules. $P(U) = \{P_1,P_2,\ldots,P_{n(i)}\}$ where $P_1 = P_{i1}/(P_{i1}+P_{i2}+\ldots+P_{in(i)})$, $P_k = P_{ik}/(P_{i1}+P_{i2}+\ldots+P_{in(i)})$, $1 \leq k \leq n(i)$.

(4) For each rule P_{ij}: $A \rightarrow a_i,\alpha_{ij}$ of the group formed for A in (1), add the rule to P:

$$\gamma \quad A^2 \rightarrow \alpha_{ij} \quad \phi \quad 1 \quad \phi \quad 1$$

where γ is the label assigned to this new rule.

(5) For each different input rule $A \rightarrow a_i$ of the group formed in (1), add the rule to P:

$$\gamma \quad A^1 \rightarrow a_i \quad S(U) \quad P(U) \quad \phi \; 1$$

where γ is the label assigned for this new rule, S(U) and P(U) are the same as defined in (3).

(6) Add the rule to P:

$$1 \quad S \rightarrow S^1 S^2 \quad S(U) \quad P(U) \quad \phi \; 1$$

where 1 is the label assigned for this new rule, $S(U) = \{\gamma(1),\gamma(2),\ldots,\gamma(m)\}$ and $P(U) = \{P_1,p_2,\ldots,P_m\}$, where m is the number of different input rules with leftside nonterminal S, $\gamma(i)$ is the label for the corresponding rule in P of the i-th input rule,

$P_i = P_{i1}+P_{i2}+\ldots+P_{in(i)}$, $n(i)$ and P_{ij} are as defined (1).

Now we proved that $L_{12} = L(G)$.

Let (xy,P) be in L_{12}. Then there exists a set of different derivation sequences. Consider one derivation sequence:

$$(S,S) \xrightarrow{P_1} (a_1A_1,\alpha_1A_1) \xrightarrow{P_2} (a_1a_2A_2,\alpha_1\alpha_2A_2)\ldots \xrightarrow{P_n} (a_1a_2\ldots a_n,\alpha_1\alpha_2\ldots\alpha_n) = (x,y).$$

From the above construction procedure of the SCFPG G, it is known that for each derivation step

$$(a_1a_2\ldots a_iA_i,\alpha_1\alpha_2\ldots\alpha_iA_i) \xrightarrow{P_i} (a_1a_2\ldots a_{i+1}A_{i+1},\alpha_1\alpha_2\ldots\alpha_{i+1}A_{i+1}),$$

there exist the production rules in P:

$$\gamma \quad A_i^1 \to a_{i+1}A_{i+1}^1 \quad S(U) \quad P(U) \quad \phi \quad 1$$

$$\gamma' \quad A_i^2 \to \alpha_{i+1}A_{i+1}^2 \quad S(U') \quad P(U') \quad \phi \quad 1$$

where $S(U)$ contains γ' and the corresponding probability in $P(U)$ is P_i/q_i for some value q_i. Besides there exists a production rule in P:

$$\gamma'' \quad A_{i-1}^2 \to \alpha_iA_i^2 \quad S(U'') \quad P(U'') \quad \phi \quad 1$$

where $S(U'')$ contains γ and the corresponding probability in $P(U'')$ is q_i. $(P_i/q_i) * q_i = P_i$.

Therefore there exists a derivation in G:

$$S \xrightarrow{1} S^1S^2 \xrightarrow{q_1} a_1A_1^1S^2 \xrightarrow{P_1/q_1} a_1A_1^1\alpha_1A_1^2 \xrightarrow{q_2} \ldots.$$

$$a_1a_2\ldots a_{i+1}A^1_{i+1}\alpha_1\alpha_2\ldots\alpha_{i+1}A^2_{i+1} \xrightarrow{q_{i+2}} \ldots \xrightarrow{P_n/q_n}$$

$$a_1a_2\ldots a_n\alpha_1\alpha_2\ldots\alpha_n = xy .$$

$$q_1\cdot(P_1/q_1)\cdot q_2\cdot(P_2/q_2)\ldots q_i\cdot(P_i/q_i)\ldots q_n\cdot(P_n/q_n)$$

$$= P_1\cdot P_2\ldots P_i\ldots P_n$$

Therefore, for each derivation of xy in T, there exists a unique corresponding derivation with the same probability in G. Besides, two different derivations of xy in T will not map to the same derivation in G. (In other words, it is a one-to-one correspondence). Similarly, for each derivation of xy in G, there exists a unique corresponding derivation with same probability in T (also a one-to-one correspondence). Therefore, there exists a one-to-one and onto relationship between the set of derivations of xy in T and the set of derivations of xy in G. Therefore,

$$(xy,P) \in L_{12} \Longleftrightarrow (xy,P) \in L(G) .$$

$$L_{12} = L(G) .$$

Theorem 2.5: Let L_1 and L_2 be the input language and the output language, respectively, of a stochastic simple syntax-directed translation schema SSSDTS $T(N,\Sigma,\Delta,R,S)$, then L_{12} is a SCFPL.

Proof: We shall construct a SCFPG G such that $L(G) = L_{12}$. A SCFPG $G(N', \Sigma', J, P, S)$ is constructed as follows:

Step 1. $N' = \{A^1 | A \in N\} \cup \{A^2 | A \in N\} \cup \{S\}$

Step 2. $\Sigma' = \Sigma \cup \Delta$.

Step 3. Since T is a SSSDTS, each rule in T is of the form

$P:\ A \to \alpha, \beta$

where $0 \leq P \leq 1$, $A \in N$, $\alpha \in (N \cup \Sigma)^*$: $\beta \in (N \cup \Delta)^*$, and α, β have the same nonterminals with the same permutation.

(1) Put all rules of R which have the same leftside nonterminal in a group within which all rules with the same input rule are further grouped together. The process is demonstrated by an example for nonterminal A as follows:

$P_{11}:\ A \to \alpha_1, \beta_{11}$

$P_{12}:\ A \to \alpha_1, \beta_{12}$

$\vdots$

$P_{1n(1)}:\ A \to \alpha_1, \beta_{1n(1)}$

$P_{21}:\ A \to \alpha_2, \beta_{21}$

$\vdots$

$P_{mn(m)}:\ A \to \alpha_m, \beta_{mn(m)}$

where $0 \leq P_{ij} \leq 1$, $\alpha_i \in (N \cup \Sigma)^*$, $\beta_{ij} \in (N \cup \Delta)^*$, α_i, β_{ij} have the same nonterminals with the same permutation, $1 \leq j \leq n(i)$, $1 \leq i \leq m$, $n(i)$ is the number of rules having the same input rule $A \to \alpha_i$, m is the number of different input rules for nonterminal A.

(2) For each rule $P_{ij}:\ A \to \alpha_i, \beta_{ij}$ of the group formed for nonterminal A in (1), where $\alpha_i = \alpha_{i1} A_1 \ldots \alpha_{in} A_n \alpha_{i(n+1)}$, $\beta_{ij} = \beta_{ij1} A_1 \ldots \beta_{ijn} A_n \beta_{ij(n+1)}$,

$\alpha_{ik} \in \Sigma^{*}$, $\beta_{ijk} \in \Delta^{*}$, $A_k \in N$, $1 \leq k \leq n+1$, $n \geq 1$,

add the rule to P:

$$\gamma \quad A^{2} \to \beta_{ij1}A_1^{2}\cdots\beta_{ijn}A_n^{2}\beta_{ij(n+1)} \qquad S(U) \quad P(U) \quad \phi \quad 1$$

where γ is the label assigned for this new rule, $S(U) = \{\gamma(1),\gamma(2),\ldots,\gamma(m)\}$, $P(U) = \{P_1,P_2,\ldots,P_m\}$, $\gamma(k)$ is the label for the corresponding rule of the k-th input rule of nonterminal A_1, $P_k = P_{k1}+P_{k2}\cdots+P_{kn(k)}$, $1 \leq k \leq m$, m is the number of different input rules of A_1, $P_{k\ell}$ is the probability of the ℓ-th translation rule of the k-th input rule for A_1, n(k) is the number of translation rules of the k-th input rule for A_1.

(3) For each different input rule $A \to \alpha_i$ of the group formed for A in (1), where $\alpha_i = \alpha_{i0}A_1\cdots\alpha_{i(n-1)}A_n\alpha_{in}$, $\alpha_{ik} \in \Sigma^{*}$, $A_k \in N$, $0 \leq k \leq n$, $n \geq 0$,

add the rule to P:

$$\gamma \quad A^{1} \to \alpha_{i0}A_1^{1}\cdots\alpha_{i(n-1)}A_n^{1}\alpha_{in} \qquad S(U) \quad P(U) \quad \phi \quad 1$$

where γ is the label assigned for this new rule, $S(U) = \{\gamma(1),\gamma(2),\ldots\gamma(n(i))\}$, $P(U) = \{P_1,P_2,\ldots,P_{n(i)}\}$, n(i) is the number of translation rules with the same input rule $A \to \alpha_i$, $\gamma(k)$ is the label of the corresponding rule in P of the k-th translation rule with input rule $A \to \alpha_i$, $P_k = P_{ik} / (P_{i1}+P_{i2}+\ldots+P_{in(i)})$, P_{ik} is the probability of the k-th translation rule, $1 \leq k \leq n(i)$.

(4) For each rule P_{ij}: $A \to \alpha_i,\beta_{ij}$ of the group formed for A in (1), where $\alpha_i \in \Sigma^{*}$,$\beta_{ij} \in \Delta^{*}$,

add the rule to P:

$$\gamma \quad A^2 \to B_{ij} \quad S(U) \quad P(U) \quad \phi \quad 1$$

where γ is the label assigned for this new rule,

$S(U) = \{\gamma(1,1),\gamma(1,2),\ldots,\gamma(1,m(1)),\gamma(2,1)\ldots\gamma(t,m(t))\}$

$P(U) = \{P_{1,1},P_{1,2},\ldots,P_{1,m(1)},P_{2,1},\ldots,P_{t,m(t)}\}$, where

t is the number of nonterminals in N, $m(k)$ is the number of different input rules for the k-th nonterminal, $\gamma(k,\ell)$ is the label of the rule added in (3) corresponding to the ℓ-th input rule of the k-th nonterminal of N, $P_{k,\ell} = P_{k,\ell,1}+P_{k,\ell,2}+\ldots+P_{k,\ell,n(k,\ell)}$, $n(k,\ell)$ is the number of translation rules associated with the ℓ-th input rule of the k-th nonterminal, $P_{k,\ell,1}$, $P_{k,\ell,2},\ldots,$ $P_{k,\ell,n(k,\ell)}$ are their corresponding probabilities.

(5) Add the rule to P:

$$\gamma \quad S \to S^1S^2 \quad S(U) \quad P(U) \quad \phi \quad 1$$

where γ is the label assigned for this new rule,

$S(U) = \{\gamma(1),\gamma(2),\ldots,\gamma(m)\}$, $P(U) = \{P_1,P_2,\ldots,P_m\}$, where

m is the number of different input rules for nonterminal S, $\gamma(i)$ is the label for the rule added in (3) corresponding to the i-th input rule of S, $P_i = P_{i1}+P_{i2}+\cdots+P_{in(i)}$, $n(i)$ is the number of translation rules associated with the i-th input rule of S, $P_{i1},P_{i2},\ldots,P_{in(i)}$ are the corresponding probabilities of these translation rules.

The proof of $L(G) = L_{12}$ is similar to that of theorem 2.4.

2.5 Illustrative Examples

Example 2.3: The following CFPG G_{SQ} generates the language

$$L_{SQ}=\{a^n b^n c^n d^n \mid n \geq 1\}$$

which could be interpreted as the language of squares of side length n = 1,2,..., [10]

$$G_{SQ} = (V_N, V_T, J, P, S)$$

where the vocabulary consists of

$$V_N = \{S, A, B, C, D\}$$

$$V_T = \{\overset{a}{\rightarrow}, b\uparrow, \overset{c}{\leftarrow}, d\downarrow\}$$

the label set is

$$J = \{1,2,3,4,5,6,7\},$$

and the production set P consists of 7 rules:

Label	Core	Success branch	Failure branch
1	$S \rightarrow aAB$	{2,3}	ϕ
2	$A \rightarrow aAC$	{2,3}	ϕ
3	$A \rightarrow D$	{4}	ϕ
4	$C \rightarrow a$	{5}	{6}
5	$D \rightarrow bDc$	{6}	ϕ
6	$B \rightarrow a$	{7}	ϕ
7	$D \rightarrow bc$	ϕ	ϕ

However, from a different point of view, this grammar can also describe the following TV pattern: $\{(x_1,x_2,x_3,x_4) | x_i$ is a line segment, x_i leads x_{i-1} by 90°, and each x_i has the same length.$\}$

Example 2.4: Zucker [56] used the concept of transformational grammar to analyze texture. He first built an ideal texture. Then a set of transformation rules were used to transform the ideal texture into a real texture. Using translation schema, we can handle these problems too. For example, consider the pattern shown in Fig. 2.1. A 2-stage tree SDTS with substitution error only can be used to characterize such a time-varying image sequence. For an iteratively varying pattern (like TV texture), an n-stage tree SDTS can be constructed to model its evolving process.

Example 2.5: With the development of CT (Computerized Tomography) technique, a 3-D object is usually represented as a sequence of slices (e.g. X-Y plane cross-sections along Z-axis). The translation schema can be applied to describe these 3-D objects. Let $\{x_1,x_2,\ldots x_n\}$ be a sequence of X-Y plane cross-sections with Z-coordinate being 1,2,...n, an n-stage SDTS can be constructed and then transformed to a CFPG. Here a simple example is given for an object described by 4 slices (Fig. 2.2).

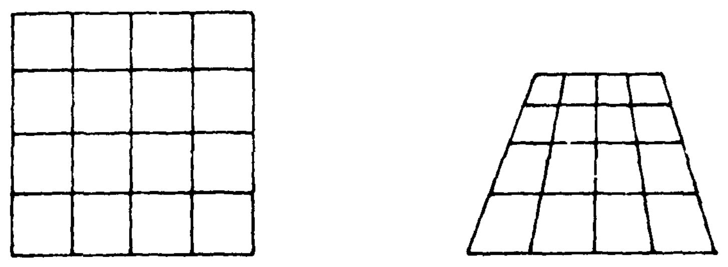


Figure 2.1 An example of tree-type transform

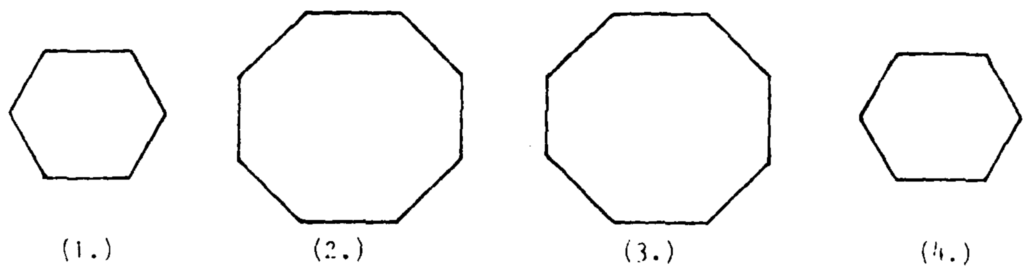


Figure 2.2 A sequence of cross sections of a 3-D object

This object is described by the following "programmed" SDTS

$$T(\{S,A,B,C,D,E,F\},\Sigma,\Sigma,\ J,\ R,\ S)$$

where $\Sigma = \{\ a\ /\ ,\ b\ |\ ,\ c\ \backslash\ ,\ d\ -\ \}$. R contains

Label	Core				Success Field
1	S→ABCDEF	,ABCDEF	,ABCDEF	,ABCDEF	{2,3}
2	A→d	,d	,d	,d	{4}
3	A→dA	,dA	,dA	,dA	{5}
4	B→a	,ab	,ab	,a	{6}
5	B→aB	,aBb	,aBb	,aB	{7}
6	C→c	,c	,c	,c	{8}
7	C→cC	,cC	,cC	,cC	{9}
8	D→d	,d	,d	,d	{10}
9	D→dD	,dD	,dD	,dD	{11}
10	E→a	,ab	,ab	,a	{12}
11	E→aE	,aEb	,aEb	,aE	{13}
12	F→c	,c	,c	,c	φ
13	F→cF	,cF	,cF	,cF	{2,3}

Example 2.6: Here an example is given to show the transformation of an SDTS into a CFPG. The transformation procedure was described in theorem 2.3.

Suppose that an SDTS is given as follows:

$$T = (N,\ \Sigma,\Sigma,R,S)$$

where N = {S,A,B,C,D}, $\Sigma = \{\ a\ \ \ ,\ b\ \ \ ,\ c\ \ \ ,\ d\ \ \ ,\ e\ \ \ \}$. R contains

S → ABCD,	ABCD,	ABCD
A → aA	, bA	, bA
A → a	, b	, b
B → dB	, eB	, eB
B → cB	, eB	, eB
B → d	, d	, d
C → cC	, aC	, bC
C → aC	, aC	, bC
C → d	, a	, b
C → dC	, aC	, bC
D → aD	, bD	, bD
D → a	, b	, b

A CFPG G(N'Σ,J,P,S) is formed as follows:

The following steps follows the procedure of theorem 2.3.

Step 1: $N' = \{A^1 | A \in N\} \cup \{A^2 | A \in N\} \cup \{A^3 | A \in N\} \cup \{S\}$

$= \{S\} \cup \{A^i, B^i, C^i, D^i, S^i | i=1,2,3\}$

Step 2: $\Sigma' = \Sigma \cup \Delta = \Sigma \cup \Sigma = \Sigma$

Step 3: (1) Add $S \rightarrow S^1S^2S^3$ { } to P. The content of the success field will be filled when all the rules are constructed.

(2) For the rule in R: S → ABCD, ABCD, ABCD

add

$S^1 \rightarrow A^1B^1C^1D^1$ { }

$S^2 \rightarrow A^2B^2C^2D^2$ { }

$S^3 \rightarrow A^3B^3C^3D^3$ { }

to P

For the rule in R: A → aA; bA, bA

add

$A^1 \rightarrow aA^1$ { }

$A^2 \rightarrow bA^2$ { }

$A^3 \rightarrow bA^3$ { }

to P

For the rule in R: A → a,b,b

add

$A^1 \rightarrow a$ { }

$A^2 \rightarrow b$ { }

$A^3 \rightarrow b$ { }

to p

and similarly for the rest of the rules of R. Then fill in those success fields as described in theorem 2.3. Then for convenience, replace A^1,B^1,C^1,D^1, A^2,B^2,C^2,D^2, A^3,B^3,C^3,D^3 by A,B,C,D,E,F,G,H,W,X,Y,Z respectively. The following is the resulting CFPG.

$N' = \{S,S^1,S^2,S^3,A,B,C,D,E,F,G,H,W,X,Y,Z\}$

P =

Label	Core	Success Field	Label	Core	Success Field
1	$S\to S^1S^2S^3$	{2}	17	$X\to d$	{18,19}
2	$S^1\to ABCD$	{3}	18	$C\to cC$	{22}
3	$S^2\to EFGH$	{4}	19	$C\to aC$	{22}
4	$S^3\to WXYZ$	{5}	20	$C\to dC$	{22}
5	$A\to aA$	{7}	21	$C\to d$	{23}
6	$A\to a$	{8}	22	$G\to aG$	{24}
7	$E\to bE$	{9}	23	$G\to a$	{25}
8	$E\to b$	{10}	24	$Y\to bY$	{18,19,20,21}
9	$W\to bW$	{5,6}	25	$Y\to b$	{26}
10	$W\to b$	{11}	26	$D\to aD$	{28}
11	$B\to dB$	{14}	27	$D\to a$	{29}
12	$B\to cB$	{14}	28	$H\to bH$	{30}
13	$B\to d$	{15}	29	$H\to b$	{31}
14	$F\to eF$	{16}	30	$Z\to bZ$	{26}
15	$F\to d$	{17}	31	$Z\to b$	ϕ
16	$X\to eX$	{11,12,13}			

Example 2.7: The stochastic translation schema $T(N,\Sigma,\Delta,R,S)$ described in Example 2.2 is transformed to a SCFPG $G(N',\Sigma',J,P,S)$ as follows:

Step 1. $N' = \{A^1 | A \in N\} \cup \{A^2 | A \in N\} \cup \{S\}$

$= \{S^1,A^1,B^1,C^1,S^2,A^2,B^2,C^2,S\}$

Step 2. $\Sigma' = \Sigma \cup \Delta = \{0,1\}$

Step 3.1. Put the translation rules of T in the order described in theorem 2.4. (They are already in order in Example 2.2).

Step 3.2. For rule $S \xrightarrow{\frac{1}{2}P_c} 0A,\ 0A$

add to P: $\gamma \quad S^2 \rightarrow 0A^2 \quad S(U) \quad P(U) \quad \phi \quad 1$

For rule $S \xrightarrow{\frac{1}{2}P_s} 0A,\ 1A$

add to P: $\gamma \quad S^2 \rightarrow 1A^2 \quad S(U) \quad P(U) \quad \phi \quad 1$

(γ , S(U) and P(U) are given later)

Other rules are added similarly.

Step 3.3. For the rules $S \xrightarrow{\frac{1}{2}P_c} 0A,\ 0A$ and $S \xrightarrow{\frac{1}{2}P_s} 0A,\ 1A$

add to P: $\gamma \ S^1 \rightarrow 0A^1 \quad S(U) \quad P(U) \quad \phi \quad 1$

Other rules are added similarly.

Step 3.4. For the rule $C \xrightarrow{\frac{1}{2}P_s} 0,\ 1$

add to P: $\gamma \ C^2 \rightarrow 1 \quad \phi \quad 1 \quad \phi \quad 1$

Other rules are added similarly.

Step 3.5. For the rules $C \xrightarrow{\frac{1}{2}P_c} 0,\ 0$ and $C \xrightarrow{\frac{1}{2}P_s} 0,\ 1$

add to P: $\gamma \ C^1 \rightarrow 0 \quad S(U) \quad P(U) \quad \phi \quad 1$

Other rules are added similarly.

Step 3.6. Add the following rule to P:

$1 \quad S \rightarrow S^1S^2 \quad S(U) \quad P(U) \quad \phi \quad 1$

The final form of the SCFPG is

$G(\{S^1,A^1,B^1,C^1,S^2,A^2,B^2,C^2,S\},\{0,1\},J,P,S)$

P:

Label	Core	U	P(U)	W	P(W)
1	$S \rightarrow S^1S^2$	{2,3}	$\{\frac{1}{2}, \frac{1}{2}\}$	ϕ	1
2	$S^1 \rightarrow 0A^1$	{10,11}	$\{P_c, P_s\}$	ϕ	1
3	$S^1 \rightarrow 1A^1$	{10,11}	$\{P_s, P_c\}$	ϕ	1
4	$A^1 \rightarrow 0B^1$	{12,13}	$\{P_c, P_s\}$	ϕ	1
5	$A^1 \rightarrow 1B^1$	{12,13}	$\{P_s, P_c\}$	ϕ	1

6	$B^1 \rightarrow 0C^1$	{14,15}	$\{P_c, P_s\}$	ϕ	1
7	$B^1 \rightarrow 1C^1$	{14,15}	$\{P_s, P_c\}$	ϕ	1
8	$C^1 \rightarrow 0$	{16,17}	$\{P_c, P_s\}$	ϕ	1
9	$C^1 \rightarrow 1$	{16,17}	$\{P_s, P_c\}$	ϕ	1
10	$S^2 \rightarrow 0A^2$	{4,5}	$\{\frac{1}{2}, \frac{1}{2}\}$	ϕ	1
11	$S^2 \rightarrow 1A^2$	{4,5}	$\{\frac{1}{2}, \frac{1}{2}\}$	ϕ	1
12	$A^2 \rightarrow 0B^2$	{6,7}	$\{\frac{1}{2}, \frac{1}{2}\}$	ϕ	1
13	$A^2 \rightarrow 1B^2$	{6,7}	$\{\frac{1}{2}, \frac{1}{2}\}$	ϕ	1
14	$B^2 \rightarrow 0C^2$	{8,9}	$\{\frac{1}{2}, \frac{1}{2}\}$	ϕ	1
15	$B^2 \rightarrow 1C^2$	{8,9}	$\{\frac{1}{2}, \frac{1}{2}\}$	ϕ	1
16	$C^2 \rightarrow 0$	ϕ	1	ϕ	1
17	$C^2 \rightarrow 1$	ϕ	1	ϕ	1

2.6 Conclusion

The time-varying pattern analysis problem is investigated in this chapter. The problem is analyzed through the use of deformation models and translation schemas. It is shown that some time-varying patterns can be characterized by a context-free programmed language. Therefore, the well developed error-correcting string parser can be applied. The proposed method can be extended to problems involving three-dimensional patterns. Tree translation is proposed to analyze more complex TV patterns. Stochastic translation is also presented to model the stochastic properties of TV patterns.

CHAPTER 3

A GENERALIZED SYNTAX-DIRECTED TREE TRANSLATION MODEL

3.1 Introduction

In chapter 2, we have proposed a syntactic method for time-varying pattern analysis. An input sequence x_1, x_2, ..., where x_i is the pattern representation at time t_i, is analyzed through a translation model which is defined as a mapping from a language L_1 to another language L_2 in formal language theory. Consider L_1 as the set of possible patterns occurring at time t_1 and L_2 as the set of patterns at time t_2. The relation governing the time-varying process of the sequence is then formulated as a translation problem. In this chapter, the formulation of string translation is extended to trees using a generalized syntax-directed model. The generalized model is compared with the conventional top-down and bottom-up tree translation model. It is shown that both the top-down and the bottom-up models are special cases of the generalized model. A parsing algorithm for this generalized tree translation model is presented.

Definitions and notations that will be referred to are briefly reviewed [57,58,59].

An alphabet Σ is ranked by a function $r:\Sigma \rightarrow N$ which assigns a rank to each member of Σ. For each n, $\Sigma_n = r^{-1}(n)$ denotes the set of symbols in Σ which have rank n. Intuitively, the rank of a symbol is the number of sons it has when it labels a node in a tree.

Let Π denote the set containing left and right brackets and comma. To avoid possible confusion, ranked alphabets are not allowed to include elements of Π. For a ranked alphabet Σ, the set $\Sigma_\star$ of (finite labeled) trees over the alphabet Σ is the least set of strings in $(\Sigma \cup \Pi)^*$ such that

(1) $\Sigma_0 \subseteq \Sigma_\star$, and

(2) for $n > 0$, $b \in \Sigma_n$, and $t_1, t_2, \ldots, t_n \in \Sigma_\star$, $b[t_1, t_2, \ldots, t_n] \in \Sigma_\star$.

<u>Definition 3.1</u>: If Σ is a ranked alphabet and Δ is an alphabet, then $\Sigma_\star(\Delta)$, the set of trees in $\Sigma_\star$ indexed by Δ, is defined recursively as follows.

1) $\Sigma_0 \cup \Delta \subseteq \Sigma_\star(\Delta)$.

2) If $b \in \Sigma_n$, $n > 0$, and $t_1, \ldots, t_n \in \Sigma_\star(\Delta)$, then $b[t_1, \ldots, t_n] \in \Sigma_\star(\Delta)$.

<u>Definition 3.2</u>: A (nondeterministic) top-down tree transducer is a 5-tuple $M = (Q, \Sigma, \Delta, R, Q_0)$ where

(1) Q is a finite set of states,

(2) Σ is a finite ranked alphabet called the input alphabet,

(3) Δ is a finite ranked alphabet called the output alphabet,

(4) $Q_0 \subseteq Q$ is a set of starting states, and

(5) R is a finite set of rules,

$$R \subseteq \bigcup_{n \geq 0} (Q \times \Sigma_n) \times (\Delta \cup (Q \times X_n))_\star.$$

A rule is written in the form $(q,b) \to w$, where $q \in Q$, $b \in \Sigma_n$, and $w \in (\Delta \cup (Q \times x_n))_*$ for some n, where X_n denotes the set $\{x_1, x_2, \ldots, x_n\}$ for $n > 0$ and X_0 denotes the empty set. x_i refers to the ith son of the current input node.

The behavior of a top-down transducer is defined inductively in terms of the output produced from a tree starting in state q.

<u>Definition 3.3</u>: Let $M = (Q, \Sigma, \Delta, R, Q_0)$ be a top-down tree transducer, and let $q \in Q$. For a tree $t \in \Sigma_*$, the set of trees output from t by M starting in state q is denoted by $M(q,t)$ and is defined inductively as follows.

(1) If $t = b \in \Sigma_0$, then $M(q,t) = \{w \mid (q,b) \to w \in R\}$;

(2) If $t = b[t_1, \ldots, t_n] \in \Sigma_*$, then

$$M(q,t) = \bigcup_{(q,b)\to w\in R} w(<p,x_j>:M(p,t_j) \mid p\in Q,\ 1 \le j \le n).$$

<u>Definition 3.4</u>: A nondeterministic bottom-up tree transducer [58] is a five-tuple $M = (Q, \Sigma, \Delta, R, F)$ where

1) Q is a finite set of states,

2) Σ and Δ are finite input and output ranked alphabets, respectively.

3) $F \subseteq Q$ is a set of final or accepting states,

4) $\Sigma \cap \Pi = \Delta \cap \Pi = \Delta \cap X = \phi$, and

5) R is a finite set of transition rules such that every rule in R is either of the form

$(b) \rightarrow (q,t)$, where $b \in \Sigma_0$, $q \in Q$, and $t \in \Delta_*$,

or of the form

$(b,q_1,\ldots,q_n) \rightarrow (q,t)$, where $n > 0$, $b \in \Sigma_n$, $q, q_1,\ldots,q_n \in Q$, and $t \in \Delta_*(X_n)$.

The behavior of a bottom-up transducer on an input tree is defined inductively as follows.

Definition 3.5: Let $M = (Q, \Sigma, \Delta, R, F)$ be a bottom-up tree transducer, and let $q \in Q$. For a tree $t \in \Sigma_*$, the set of trees which M can output from t ending in state q is denoted by $M(q,t)$ and is defined inductively as follows:

(1) For $b \in \Sigma_0$, $M(q,b) = \{w \mid b \rightarrow (q,w) \in R\}$,

(2) For $t = b[t_1,\ldots,t_n] \in \Sigma_*$,

$M(q,t) = \{v \mid$ for some rule $(b,q_1,\ldots,q_n) \rightarrow (q,w) \in R$,

some i, $1 \leq i \leq n$, and some $u_i \in M(q_i,t_i)$, $v \in W(x_i:u_i)\}$

Definition 3.6: A generalized syntax directed translation (GSDT) [57] is a four-tuple $F = (G, \Delta, \Gamma, R)$, where:

(1) $G = (V, \Sigma, P, S)$ is a proper context free grammar;

(2) Δ is a finite set of output symbols;

(3) Γ is a finite set of distinct translation symbols of the form $\tau_i(A)$, where i is an integer and A is in $V - \{S\}$, plus the symbol S_1. Whenever it is possible to do so without confusion, we will denote $\tau_i(A)$ by A_i. We call A_i the i-th translation symbol associated with A.

(4) R is a function which associates with each production $A \rightarrow \alpha$ in P, a set of semantic rules $\{A_1 = \beta_1, A_2 = \beta_2,\ldots,A_m = \beta_m\}$, in which each

β_i is a string in $(\Gamma \cup \Delta)^*$, such that all translation symbols appearing in β_i are translation symbols associated with nonterminals appearing in α.

For each x in Σ^* we define $F(x)$, the set of outputs of x as follows:

(1) If x is not in $L(G)$, then $F(x) = \phi$.

(2) If x is in $L(G)$, then each parse tree with yield x defines an element y in $F(x)$, which is the value of the translation symbol S_1 associated with the root. The value of S_1 is computed bottom-up as follows:

(i) With each interior node N of the parse tree labeled $A \to \alpha$ are associated the translation symbols $A_1, A_2, \ldots, A_m$, which are all the translation symbols associated with A. The values of these translation symbols at N are computed using the semantic rules and the values of the translation symbols at the descendants of N as follows.

(ii) Suppose α is $x_0B_1x_1B_2x_2 \ldots B_kx_k$, where x_j is in Σ^* and B_j is in V, $0 \leq j \leq k$. Suppose $A_i = y_0C_1y_1C_2y_2 \ldots C_ly_l$ is the semantic rule for A_i, where y is in Δ^* and C_j is a translation symbol in Γ associated with B_{h_j} for some $1 \leq h_j \leq k$. Then $v(A_i)$, the value of A_i at node N, is the string $y_0v(C_1)y_1v(C_2)y_2 \ldots v(C_l)y_l$ in Δ^*, where $v(C_j)$ is the value of C_j at the descendant of N which is labeled by a B_{h_j} production.

T(F), the translation defined by F, is the set $\{(x,y) | y \in F(x)\}$.

<u>Example 3.1</u>: Let $F = (G, \{a,b\}, \{S_1, A_1, A_2, B_1, B_2\}, R)$, where the productions of the grammar and the associated semantic rules are:

Productions Semantic rules

(1)	$S \rightarrow A$	$S_1 = A_1A_2$
(2)	$A \rightarrow aAbB$	$A_1 = aA_1B_1$
		$A_2 = bA_2B_2$
(3)	$A \rightarrow bAaB$	$A_1 = aA_1B_1$
		$A_2 = bA_2B_2$
(4)	$B \rightarrow A$	$B_1 = A_1$
		$B_2 = A_2$
(5)	$A \rightarrow \varepsilon$	$A_1 = \varepsilon$
		$A_2 = \varepsilon$

F defines the translation $\{(w,a^ib^i) | i \geq 0$ and $w \in \{a,b\}^*$, such that w has i a's and i b's$\}$.

3.2 Generalized Syntax-Directed Tree Translation

A top-down tree transducer (TDTT) does not specify an input language, it assumes that all trees in Σ_* are input sentences. A bottom-up tree transducer (BUTT) does specify an input language, which is a subset of Σ_*, but given an input tree sentence, even though a BUTT can copy a subtree in an output sentence many times, all the copies of a subtree are the same, while a TDTT can generate different copies of a subtree. There exist translations which can be implemented by a TDTT, but not by a BUTT. There also exist translations which can be implemented by a BUTT, but not by a TDTT.

Example 3.2: An arithmetic expression involving addition, multiplication, a constant c, and a variable y may be represented by a tree over the alphabet $\Sigma = \{+, *, y, c\}$, where + and * have rank 2 and y and c have rank 0. We construct a deterministic top-down transducer M which takes the formal derivative with respect to y of the expression represented by an input tree in Σ_*. Let $\Delta = \Sigma \cup \{1,0\}$ where 1 and 0 have rank 0. Let $M = (\{D,I\}, \Sigma, \Delta, R, \{D\})$ be a top-down transducer, where R contains the following rules:

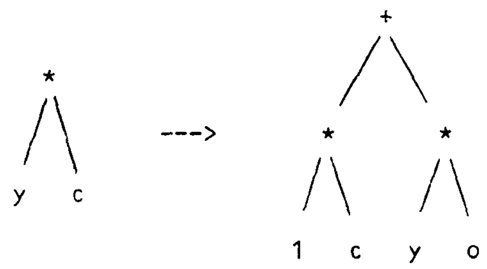
$(D,+) \rightarrow +[<D,x_1>,<D,x_2>]$,

$(D,*) \rightarrow +[*[<D,x_1>,<I,x_2>], *[<I,x_1>,<D,x_2>]]$,

$(D,y) \rightarrow 1$, $(D,c) \rightarrow 0$, $(I,\sigma) \rightarrow \sigma$ for $\sigma \in \{y,c\}$, and

$(I,\sigma) \rightarrow c[<I,x_1>,<I,x_2>]$ for $\sigma \in \{+,*\}$.

The tree translation defined by M cannot be defined by a BUTT. The main reason is due to the second rule above. With this rule, the subtree x_1 of an input tree $*[x_1,x_2]$ is translated two times as $<D,x_1>$ and $<I,x_1>$ respectively. For instance,



In the above translation, y is translated two times differently (the first time as 1 and the second time as y). But from the definition of BUTT, we know that BUTT cannot generate this tree translation pair. On the other hand, a translation which contains only a single tree pair

```
  *     +
{( /\ ,  /\ )} cannot be implemented by a TDTT because a TDTT cannot
  y c   r o
```

specify finite input tree set. While this translation can be implemented by the following BUTT:

$$M=(\{q_0,q_1,q_2\}, \{y,c,*\}, \{1,0,+\}, R, \{q_0\})$$

$$R=\{y\rightarrow(q_1,1), c\rightarrow(q_2,0), (*,q_1,q_2)\rightarrow(q_0, +[x_1,x_2])\}.$$

Based on the definition of GSDT given by Aho and Ullman, we propose a similar definition for trees, called the generalized syntax-directed tree translation (GSDTT). It can be shown that both TDTT and BUTT are special cases of GSDTT.

A GSDTT is defined as follows:

Definition 3.7: A generalized syntax-directed tree translation (GSDTT) is a four-tuple $F = (G, \Delta, \Gamma, R)$, where:

(1) $G = (V, \Sigma, P, S)$ is a regular tree grammar;

(2) Δ is a finite set of output symbols;

(3) Γ is a finite set of distinct translation symbols of the form $\tau_i(A)$, where i is an integer and A is in $V - \{S\}$, plus the symbol S_1. Whenever it is possible to do so without confusion, we will denote $\tau_i(A)$ by A_i. We call A_i the i-th translation symbol associated with A.

(4) R is a function which associates with each production $A \rightarrow \alpha$ in P, a set of semantic rules $\{A_1 = \beta_1, A_2 = \beta_2, \ldots, A_m = \beta_m\}$, in which each β_i is a tree in $(\Gamma \cup \Delta)_*$, such that all translation symbols appearing in β_i are translation symbols associated with nonterminals appearing in α.

For each x in Σ^* we define $F(x)$, the set of outputs of x as follows:

(1) If x is not in $L(G)$, then $F(x) = \phi$.

(2) If x is in L(G), then each parse tree with yield x defines an element y in F(x), which is the value of the translation symbol S_1 associated with the root. The value of S_1 is computed bottom-up as follows:

(i) With each interior node N of the parse tree labeled $A \rightarrow \alpha$ are associated the translation symbols $A_1, A_2, \ldots, A_m$, which are all the translation symbols associated with A. The values of these translation symbols at N are computed using the semantic rules and the values of the translation symbols at the descendants of N as follows.

(ii) Suppose $\{B_j | 0 \leq j \leq k$ for some $k\}$ is the set of nonterminals appearing in α, $A_i = \beta_i$ is the semantic rule for A_i and $\{C_j | 1 \leq j \leq k$ for some $l\}$ is the set of translation symbols appearing in β_i with C_j being the translation symbol associated with B_{h_j}. Then $v(A_i)$, the value of A_i at node N, is the tree β_i with each C_j being replaced by $v(C_j)$, where $v(C_j)$ is the value of C_j at the descendant of N which is labeled by a B_{h_j} production. T(F), the translation defined by F, is the set $\{(x,y) | y \in F(x)\}$.

Theorem 3.1: For any BUTT $M = (Q, \Sigma, \Delta, R, F)$ there exists a GSDTT $F = (G, \Delta', \Gamma, R')$ such that M and F define the same translation.

Proof: Given $M=(Q, \Sigma, \Delta, R, F)$ we can construct $F=(G, \Delta', \Gamma, R')$ as follows: Let $\Delta' = \Delta$, and $G = (V, \Sigma', P, S)$ where $\Sigma' = \Sigma$. $V=\{A_q | q \in Q\}$, $S = A_{q_0}$, q_0 is the accepting state and P and R' are formed as follows: For each $(b) \rightarrow (q,t)$ in R where $b \in \Sigma_0$, $q \in Q$; $t \in \Delta_*$ add $A_q \rightarrow b$, $A_{q1} \rightarrow t$ to P, R' respectively. For each $(b, q_1, \ldots, q_n) \rightarrow (q,t)$ where $n>0$, $b \in \Sigma_n$, $q_i \in Q$, $t \in \Delta_*(X_n)$ add $A_q \rightarrow b[A_{q_1}, A_{q_2}, \ldots, A_{q_n}]$, $A_{q_1} \rightarrow t'$ to P, R' where t' is the same as t except that each X_i in t is replaced by

A_{q_i}.

The next thing is to show that M and F define the same translation: If $<b,t> \in M$, $b \in \Sigma_0$, $t \in \Delta_*$, it can be found straightforward from the construction procedure of F that $<b,t> \in F$.

If $<r,t> \in M$, the depth of r is k and if the statement that $<r',t'> \in M \Rightarrow <r',t'> \in F$ holds for all $<r',t'>$ where the depth of r' is K-1, then suppose $r=b[r_1, r_2, \ldots, r_n]$ where $b \in \Sigma_n$, $r_i \in \Sigma_*$, the depth of $r_i \leq K-1$. Since $<r,t> \in M$, there exists a rule in R of the form $(b, q_1, \ldots, q_n) \rightarrow (q_0, u)$ where q_0 is accepting state and $u \in \Delta_*(X_n)$, besides, t is equal to the tree of u after each X_i in u being replaced by u_i where $u_i \in M(q_i,r_i)$, then there exists a rule in $R':A_{q_{01}} \rightarrow u$ with each x_i of u being replaced by $A_{q_{i1}}$ and a corresponding rule in P : $A_{q_0} \rightarrow b[A_{q_1}, A_{q_2}, \ldots, A_{q_n}]$, and $A_{q_i} \rightarrow r_i$ (* by theorem in Brainerd [59]) the depth of all $r_i < k$, therefore by induction we know that $<r,t> \in M \Rightarrow <r,t> \in F$. Similarly, we can show that $<r,t> \in F \Rightarrow <r,t> \in M$ therefore M and F define the same translation .

Theorem 3.2: For any TDTT $M=(Q, \Sigma, \Delta, R, Q)$ there exists a GSDTT $F=(G, \Delta', \Gamma, R')$ such that M and F define the same translation.

Proof: We prove it by constructing a GSDTT $F=(G, \Delta', \Gamma, R')$ as follows: Let $G=(V, \Sigma', P, S)$, $\Sigma' = \Sigma$, $\Delta' = \Delta$, such that $L(G) = \Sigma_*$ (* the existence of G is proved in Brainerd [59]). Let $\Gamma = V \times Q$ i.e. suppose the number of states in Q is m, then $A_1, \ldots, A_m \subset \Gamma$ for $A \in V$. For each rule in R :$(q,b) \rightarrow w$, where $q \in Q$, $b \in \Sigma_n$, $w \in \Delta_*(Q \times X_n)$ add to R': $A_q \rightarrow u$, where u is the tree of w after each $q'X_i$ being replaced by $B_{iq'}$ corresponding to each rule $A \rightarrow b[B_1, B_2, \ldots, B_n]$ in P. Next step is to prove that $<r,t> \in M \Longleftrightarrow <r,t> \in F$: If $<r,t> \in M$ and $r=b$, $b \in \Sigma_0$, then there exists a rule in R : $(q_0,b) \rightarrow t$ therefore there exists a rule in P : $S \rightarrow b$, and a rule in R': $S_{q_0} \rightarrow t$ so $<b,t> \in F$. If $<r,t> \in M$ and $r=b[r_1, r_2, \ldots, r_n]$, $b \in \Sigma_n$ then there exist a rule in R : $(q_0,b) \rightarrow t'$ where $t' \in \Delta_*(Q \times X_n)$ and the terminal part of t' can be matched with t. Therefore there exists a rule in P : $S \rightarrow b[B_1, B_2, \ldots, B_n]$ and a rule in R': $S_{q_0} \rightarrow u$ where $B_i \in V$, u is the result of replacing each q_{x_i} in t' by B_{iq}. If the statement that $<r,t> \in M \Rightarrow <r,t> \in F$ is true for all r of depth K-1 then it's also true for all r of depth K. We already know that it's true for K=1 therefore by induction we know that $<r,t> \in M \Rightarrow <r,t> \in F$. Similarly, it can be shown that $<r,t> \in F \Rightarrow <r,t> \in M$. So, M and F define the same translation.

Example 3.3: Let $\Sigma = \{+,\cdot,y,c\}$ be ranked alphabet, where + and $\cdot$ have rank of 2, and y and c have rank 0. We construct an NGSDT G which takes the formal derivative with respect to y of the expressions represented by input trees in Σ_*, where c represents a constant. Let $\Delta = \{+,\cdot,1,0,y,[,]\}$. Let $G=(\{d,I\},\Sigma,\Delta,R,\{d\})$, where

$$\begin{aligned}
R=\{ & (d,+)\rightarrow[(d,x_1)+(d,x_2)], \\
& (d,\cdot)\rightarrow[[(d,x_1)\cdot(I,x_2)]+[(I,x_1)\cdot(d,x_2)]], \\
& (d,y)\rightarrow 1, (d,c)\rightarrow 0\} \cup \\
& \{(I,\sigma)\rightarrow\sigma \mid \sigma\in\{y,c\}\} \cup \\
& \{(I,\sigma)\rightarrow[(I,x_1)\sigma(I,x_2)] \mid \sigma\in\{+,\cdot\}\}.
\end{aligned}$$

This TDTT can be replaced by a GSDTT:

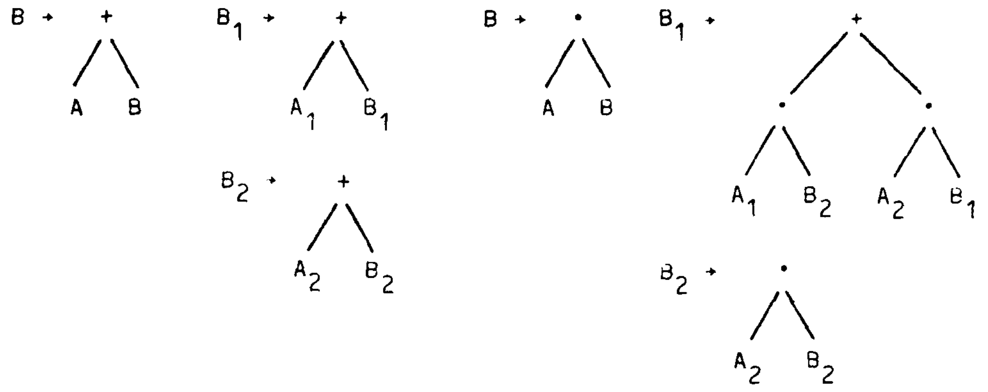
$F=(G, \Delta, \Gamma, R)$ $G=(V, \Sigma, P, S)$

P:	R:	P:	R:
$S \rightarrow y$	$S_1 \rightarrow 1$	$S \rightarrow c$	$S_1 \rightarrow 0$
$A \rightarrow y$	$A_1 \rightarrow 1$	$A \rightarrow c$	$A_1 \rightarrow 0$
	$A_2 \rightarrow y$		$A_2 \rightarrow c$
$B \rightarrow y$	$B_1 \rightarrow 1$	$B \rightarrow c$	$B_1 \rightarrow 0$
	$B_2 \rightarrow y$		$B_2 \rightarrow c$

$$S \rightarrow \begin{tikzcd} & + \arrow[dl, dash] \arrow[dr, dash] & \\ A & & B \end{tikzcd} \qquad S_1 \rightarrow \begin{tikzcd} & + \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_1 \end{tikzcd} \qquad S \rightarrow \begin{tikzcd} & \cdot \arrow[dl, dash] \arrow[dr, dash] & \\ A & & B \end{tikzcd} \quad S_1 \rightarrow \begin{tikzcd} & & & + \arrow[dll, dash] \arrow[drr, dash] & & & \\ & \cdot \arrow[dl, dash] \arrow[dr, dash] & & & & \cdot \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_2 & & A_2 & & B_1 \end{tikzcd}$$

$$A \rightarrow \begin{tikzcd} & \cdot \arrow[dl, dash] \arrow[dr, dash] & \\ A & & B \end{tikzcd} \qquad A_1 \rightarrow \begin{tikzcd} & + \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_1 \end{tikzcd} \qquad A \rightarrow \begin{tikzcd} & + \arrow[dl, dash] \arrow[dr, dash] & \\ A & & B \end{tikzcd} \quad A_1 \rightarrow \begin{tikzcd} & & & + \arrow[dll, dash] \arrow[drr, dash] & & & \\ & \cdot \arrow[dl, dash] \arrow[dr, dash] & & & & \cdot \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_2 & & A_2 & & B_1 \end{tikzcd}$$

$$A_2 \rightarrow \begin{tikzcd} & \cdot \arrow[dl, dash] \arrow[dr, dash] & \\ A_2 & & B_2 \end{tikzcd} \qquad\qquad A_2 \rightarrow \begin{tikzcd} & + \arrow[dl, dash] \arrow[dr, dash] & \\ A_2 & & B_2 \end{tikzcd}$$



Algorithm 3.1: Parsing for GSDTT.

Input. A GSDTT $F=(G, \Delta, \Gamma, R)$, $G=(V, \Sigma, P, S)$ and an input tree pair $\langle r,t \rangle$

Output. All the correct parses for $\langle r,t \rangle$. If no parse exists output "error".

Method.

(1) For each nonterminal A in V, order the alternates in P for A. Let A_{io} be the index for the ith alternate of A. Let A_{ij} be the index for the jth alternate in R with respect to A_{io} in P.

(2) Let $\langle u,v \rangle$ be a new tree pair. A 4-tuple (X, b, l, K) will be used to denote the configuration of each *node in a tree:*

(a) X denotes the nonterminal of the node before it is rewritten.

(b) b denotes the terminal of the node after it is rewritten.

(c) l denotes the label of the production rule for rewriting the node.

(d) k denotes the sequence number for counting the sequence of rule applications.

(3) let $X = S$, $i=1$, $<u,v> = <(S, \ , 0, 1), (S_1, \ , 0, 1)>$.

(4) Select the first rule of P for X. $j=1$.

(5) replace node $(X, \ , l, k)$ of u with rule X_{jo} in P. If X_{jo} is $X \rightarrow b[B_1, B_2, \ldots, B_m]$, then replace the node with $(X, b, j, i)[(B_1, \ , 0, i), \ldots, (B_m, \ , 0, i)]$.

(6) Check compatibility for u and r, if compatible go to (7) else go to (11).

(7) For each node $(X_n, \ , 0, k)$ of v, where n is any number, replace it with rule X_{jn} in R. If a node in X_{jn} is a terminal b, it is written as (X_n, b, j, i), if a node in X_{jn} is a nonterminal B_m, it is written as $(B_m, \ , 0, i)$, go to (8).

(8) Check compatibility for v and t, if compatible, go to (9), else (12).

(9) Check if there is any nonterminal node left in u, if yes, let X be the next *nonterminal*, $i=i+1$, go to (4), if not, go to (10).

(10) Check compatibility for u and r, v and t, if compatible, find the parse by tracing down tree u, report success and go to (12), if not, go to (12).

(11) delete the subtree just being added to u, restore its root with $(X, \ , j, k)$. go to (13).

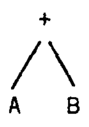
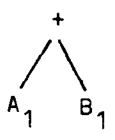
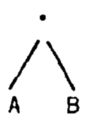
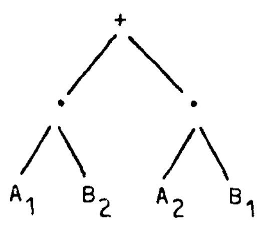
(12) delete the subtree just being added to u, restore its root with $(X, \ , j, k)$, delete the subtrees being added to v, restore their roots with $(X_n, \ , j, k)$ go to (13).

(13) check if there are new rules available for X, if yes, set $j=j+1$.

go to (5), if not go to (14).

(14) If X=S, then finished, exit. If not, then backtrack, take the immediate father of node (X, , j, k) as new X, i=i-1, and go to (4) (Figure 3.1 shows the flowchart for this algorithm)

Example 3.4: The translation rules of the GSDTT defined in example 3.3 is labeled, reordered and written as follows:

P:	R:
$(S_{10})\ S \to y$	$(S_{11})\ S_1 \to 1$
$(S_{20})\ S \to C$	$(S_{21})\ S_1 \to 0$
$(S_{30})\ S \to$ 	$(S_{31})\ S_1 \to$ 
$(S_{40})\ S \to$ 	$(S_{41})\ S_1 \to$ 
$(A_{10})\ A \to y$	$(A_{11})\ A. \to 1$ $(A_{12})\ A_2 \to y$
$(A_{20})\ A \to C$	$(A_{21}\ A_1 \to 0$ $(A_{22})\ A_2 \to C$

$(A_{30})\ A \to$
$$\begin{tikzcd}[column sep=tiny] & + \arrow[dl, no head] \arrow[dr, no head] & \\ A & & B \end{tikzcd}$$

$(A_{31})\ A_1 \to$
$$\begin{tikzcd}[column sep=tiny] & + \arrow[dl, no head] \arrow[dr, no head] & \\ A_1 & & B_1 \end{tikzcd}$$

$(A_{32})\ A_2 \to$
$$\begin{tikzcd}[column sep=tiny] & + \arrow[dl, no head] \arrow[dr, no head] & \\ A_2 & & B_2 \end{tikzcd}$$

$(A_{32})\ A_2 \to$
$$\begin{tikzcd}[column sep=tiny] & + \arrow[dl, no head] \arrow[dr, no head] & \\ A_2 & & B_2 \end{tikzcd}$$

$(A_{40})\ A \to$
$$\begin{tikzcd}[column sep=tiny] & \cdot \arrow[dl, no head] \arrow[dr, no head] & \\ A & & B \end{tikzcd}$$

$(A_{41})\ A_1 \to$
$$\begin{tikzcd}[column sep=tiny] & & & + \arrow[dll, no head] \arrow[drr, no head] & & & \\ & \cdot \arrow[dl, no head] \arrow[dr, no head] & & & & \cdot \arrow[dl, no head] \arrow[dr, no head] & \\ A_1 & & B_2 & & A_2 & & B_1 \end{tikzcd}$$

$(A_{42})\ A_2 \to$
$$\begin{tikzcd}[column sep=tiny] & \cdot \arrow[dl, no head] \arrow[dr, no head] & \\ A_2 & & B_2 \end{tikzcd}$$

$(B_{10})\ B \to y$ $\quad$ $(B_{11})\ B_1 \to 1$ $\quad$ $(B_{12})\ B_2 \to y$

$(B_{20})\ B \to C$ $\quad$ $(B_{21})\ B_1 \to 0$ $\quad$ $(B_{22})\ B_2 \to C$

$(B_{30})\ B \to$
$$\begin{tikzcd}[column sep=tiny] & + \arrow[dl, no head] \arrow[dr, no head] & \\ A & & B \end{tikzcd}$$

$(B_{31})\ B_1 \to$
$$\begin{tikzcd}[column sep=tiny] & + \arrow[dl, no head] \arrow[dr, no head] & \\ A_1 & & B_1 \end{tikzcd}$$

$(B_{32})\ B_2 \to$
$$\begin{tikzcd}[column sep=tiny] & + \arrow[dl, no head] \arrow[dr, no head] & \\ A_2 & & B_2 \end{tikzcd}$$

$(B_{40})\ B \to$
$$\begin{tikzcd}[column sep=tiny] & \cdot \arrow[dl, no head] \arrow[dr, no head] & \\ A & & B \end{tikzcd}$$

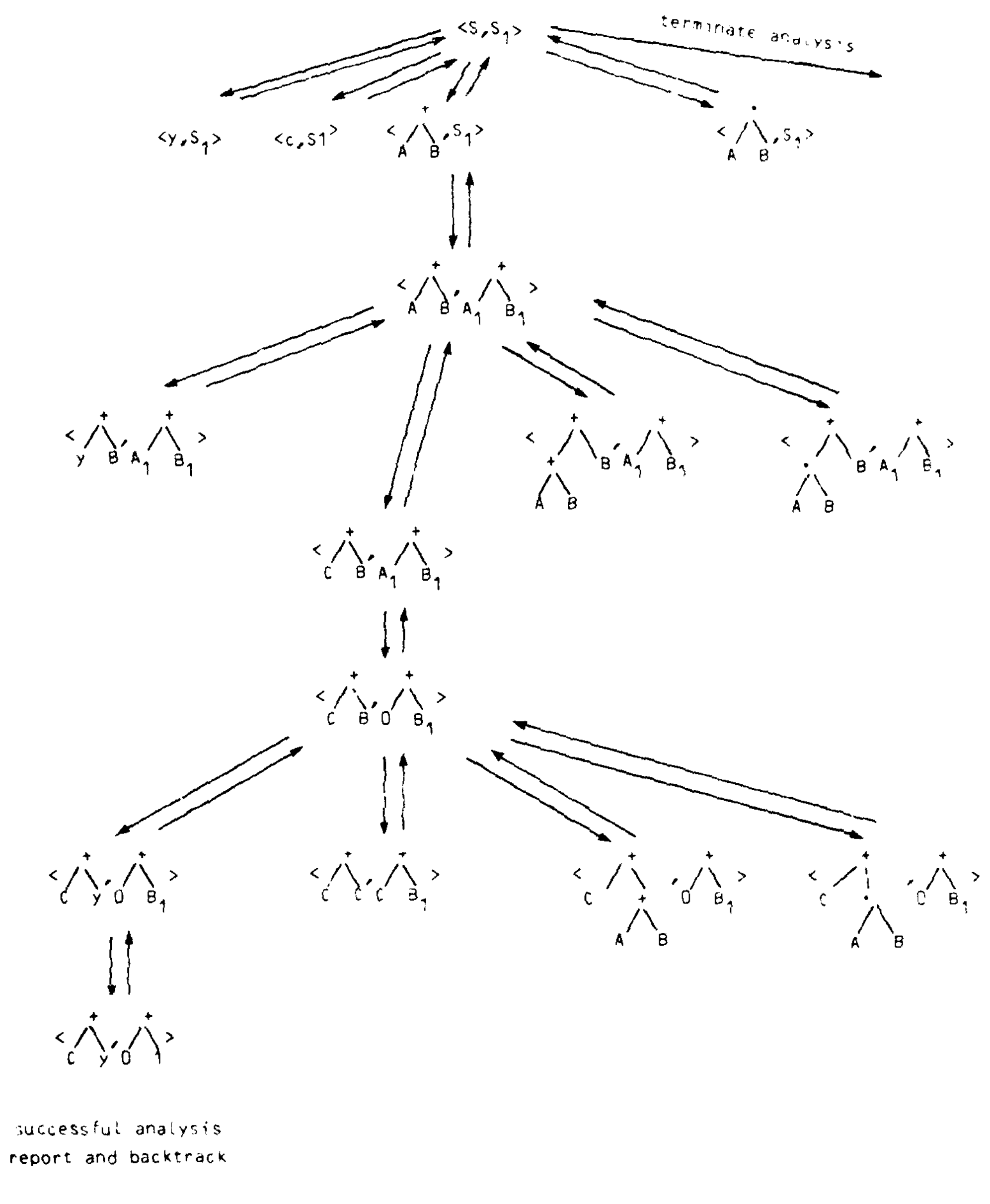
$(B_{41})\ B_1 \to$
$$\begin{tikzcd}[column sep=tiny] & & & + \arrow[dll, no head] \arrow[drr, no head] & & & \\ & \cdot \arrow[dl, no head] \arrow[dr, no head] & & & & \cdot \arrow[dl, no head] \arrow[dr, no head] & \\ A_1 & & B_2 & & A_2 & & B_1 \end{tikzcd}$$

$(B_{42})\ B_2 \to$
$$\begin{tikzcd}[column sep=tiny] & \cdot \arrow[dl, no head] \arrow[dr, no head] & \\ A_2 & & B_2 \end{tikzcd}$$

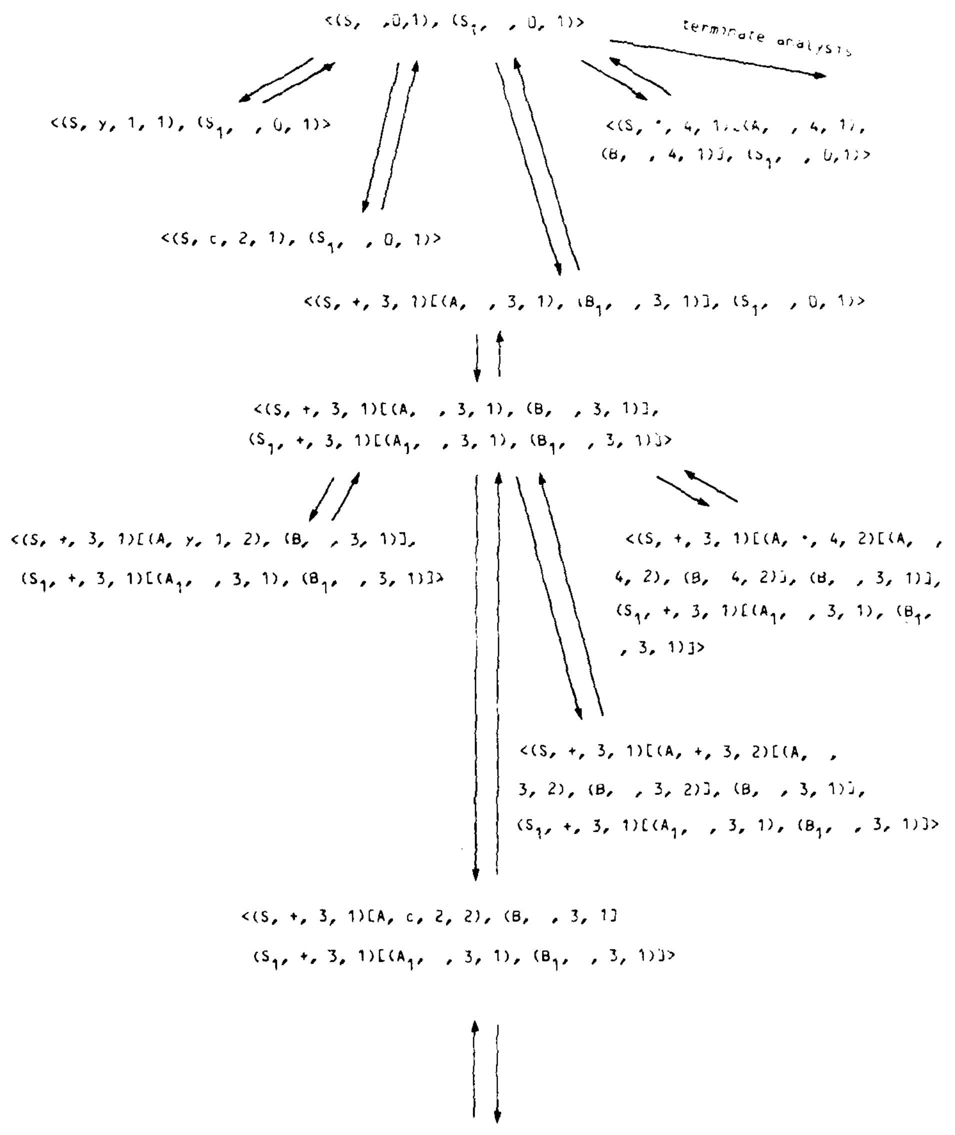
The following diagram shows the analysis of the input tree pair

$$\begin{tikzcd}[column sep=tiny] & & + \arrow[dl, no head] \arrow[dr, no head] & & & & + \arrow[dl, no head] \arrow[dr, no head] & & \\ < & c & & y & , & 0 & & 1 & > \end{tikzcd}$$

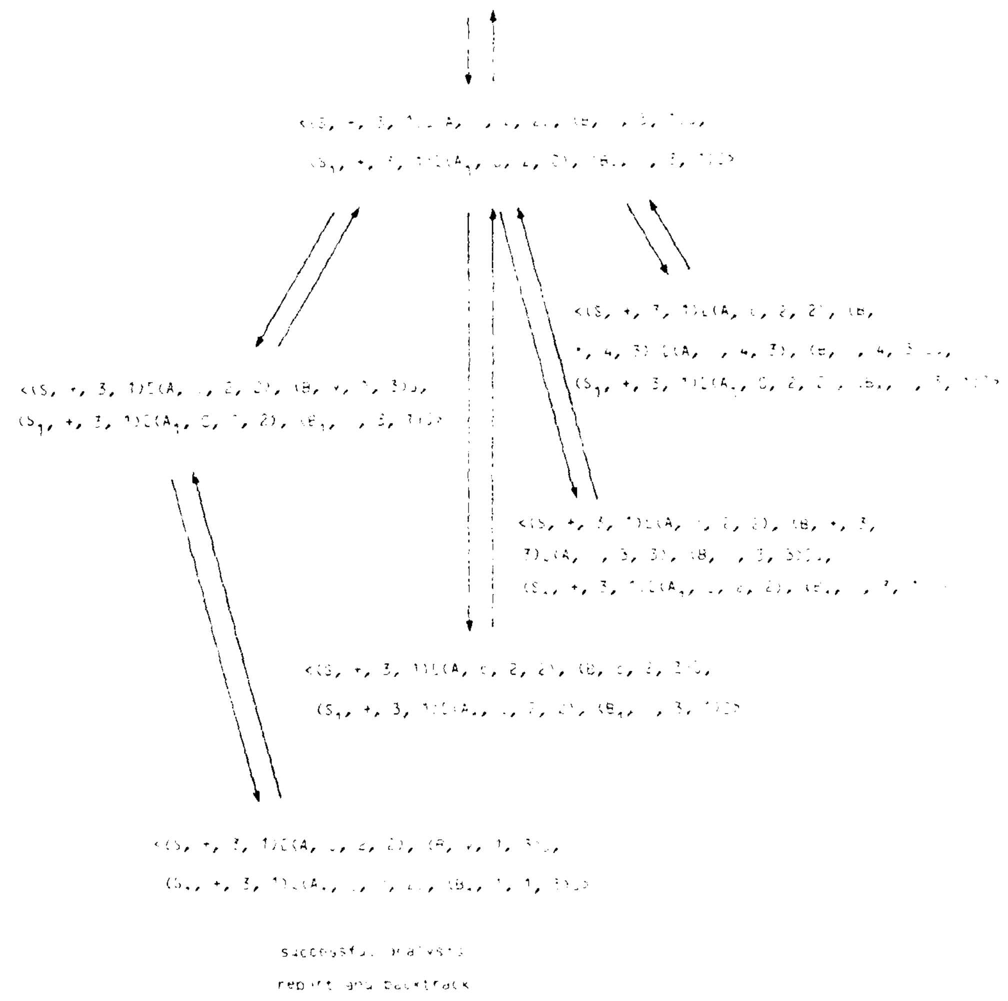
with respect to this GSDTT:



A more detailed diagram containing information about each node's configuration is shown as follows:



<(S, ,0,1), (S1, , 0, 1)>
terminate analysis
<(S, y, 1, 1), (S1, , 0, 1)>
<(S, *, 4, 1)[(A, , 4, 1), (B, , 4, 1)], (S1, , 0,1)>
<(S, c, 2, 1), (S1, , 0, 1)>
<(S, +, 3, 1)[(A, , 3, 1), (B1, , 3, 1)], (S1, , 0, 1)>
<(S, +, 3, 1)[(A, , 3, 1), (B, , 3, 1)], (S1, +, 3, 1)[(A1, , 3, 1), (B1, , 3, 1)]>
<(S, +, 3, 1)[(A, y, 1, 2), (B, , 3, 1)], (S1, +, 3, 1)[(A1, , 3, 1), (B1, , 3, 1)]>
<(S, +, 3, 1)[(A, *, 4, 2)[(A, , 4, 2), (B, 4, 2)], (B, , 3, 1)], (S1, +, 3, 1)[(A1, , 3, 1), (B1, , 3, 1)]>
<(S, +, 3, 1)[(A, +, 3, 2)[(A, , 3, 2), (B, , 3, 2)], (B, , 3, 1)], (S1, +, 3, 1)[(A1, , 3, 1), (B1, , 3, 1)]>
<(S, +, 3, 1)[A, c, 2, 2), (B, , 3, 1] (S1, +, 3, 1)[(A1, , 3, 1), (B1, , 3, 1)]>



<(S, +, 3, 1)[illegible]
(S1, +, 3, 1)[(A1, [illegible], 2), (B1, [illegible], 3, 1)]>
<(S, +, 3, 1)[(A, [illegible], 2, 2), (B, +, 3, 3)],
(S1, +, 3, 1)[(A1, 0, 1, 2), (B1, [illegible], 3, 1)]>
<(S, +, 3, 1)[(A, [illegible], 2, 2), (B,
+, 4, 3)[(A, [illegible], 4, 3), (B, [illegible], 4, 3)]],
(S1, +, 3, 1)[(A1, 0, 2, [illegible]), (B1, [illegible], 3, 1)]>
<(S, +, 3, 1)[(A, [illegible], 2, 2), (B, +, 3,
3)[(A, [illegible], 3, 3), (B, [illegible], 3, 3)]],
(S1, +, 3, 1)[(A1, [illegible], 2, 2), (B1, [illegible], 3, 1)]>
<(S, +, 3, 1)[(A, [illegible], 2, 2), (B, [illegible], 3, 3)],
(S1, +, 3, 1)[(A1, [illegible], 2, 2), (B1, [illegible], 3, 1)]>
<(S, +, 3, 1)[(A, [illegible], 2, 2), (B, +, 1, 3)],
(S1, +, 3, 1)[(A1, [illegible], 2), (B1, [illegible], 1, 3)]>
successful analysis
report and backtrack

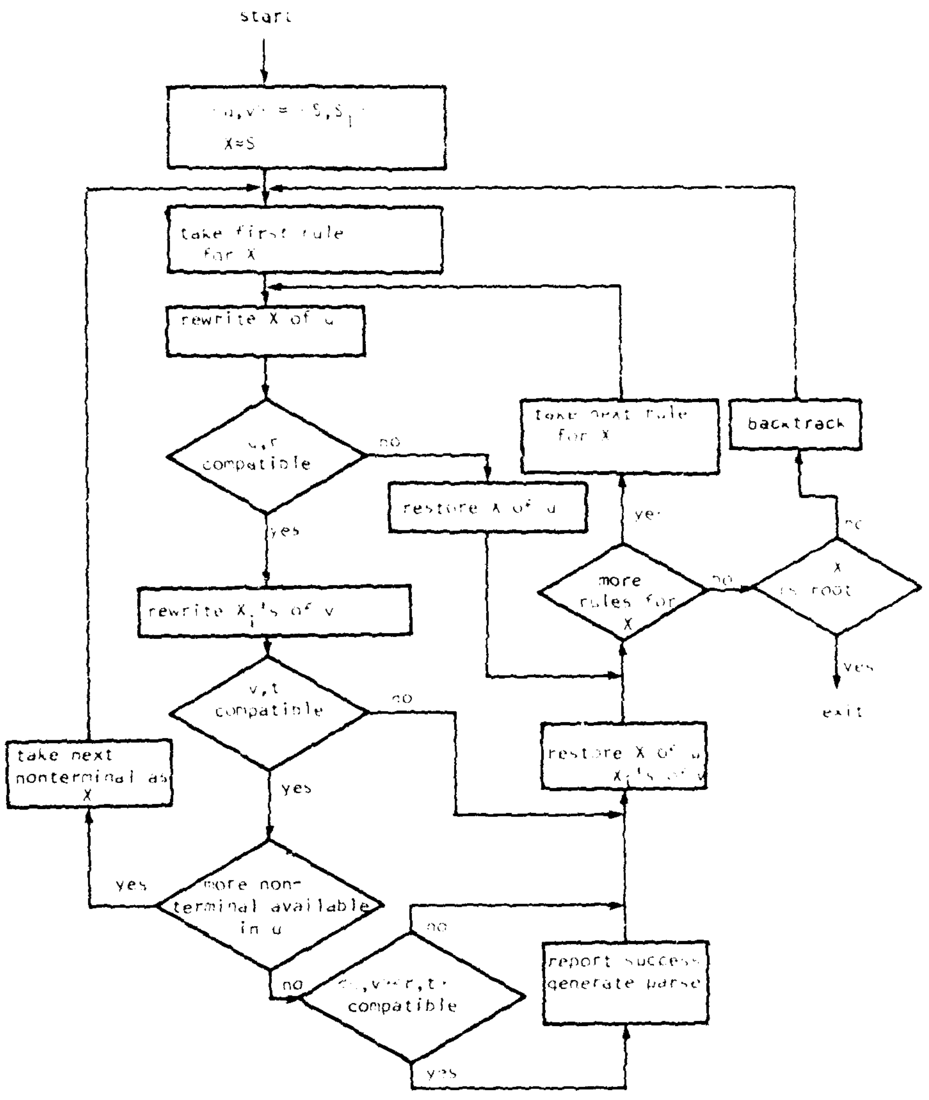


Figure 3.1 Flow chart for the parsing of a GSDTT

3.3 Stochastic Generalized Syntax-Directed Tree Translation

Definition 3.8: A stochastic generalized syntax-directed tree translation (SGSDTT) is a four-tuple $F = (G, \Delta, \Gamma, R)$, where:

(1) $G = (V, \Sigma, P, S)$ is a regular tree grammar;

(2) Δ is a finite set of output symbols;

(3) Γ is a finite set of distinct translation symbols of the form $\tau_i(A)$, where i is an integer and A is in $V - \{S\}$, plus the symbol S_1. whenever it is possible to do so without confusion, we will denote $\tau_i(A)$ by A_i. We call A_i the i-th translation symbol associated with A.

(4) R is a function which associates with a probability value p, $0 \leq p \leq 1$ each production $A \rightarrow \alpha$ in P, and a set of semantic rules $\{A_1 = \beta_1, A_2 = \beta_2, \ldots, A_m = \beta_m\}$, in which each β_i is a tree in $(\Gamma \cup \Delta)_*$, such that all translation symbols appearing in β_i are translation symbols associated with nonterminals appearing in α.

Definition 3.9: An SGSDTT is:

(a) unrestricted if for each rule $p: A \rightarrow \alpha, \beta$ (β is the set of semantic rules associated with $A \rightarrow \alpha$) the probability p is not conditioned on other rules or events.

(b) proper if for each nonterminal A the probabilities of all rules in which A is the leftside nonterminal sum to 1.

We consider only unrestricted, proper translation.

Definition 3.10: The stochastic translation from $\Sigma_\star$ to $\Delta_\star$ produced by F with starting symbol S is the set

$$f(F,S) = \{(x,y,p) \mid x \text{ in } \Sigma_*,\ y \text{ in } \Delta_*,\ P = \sum_{i=1}^{n} \prod_{j=1}^{f(i)} p_{ij}(x,y)\}$$

where there are n distinct standard derivations (e.g., breadth-first) of (x,y) with f(i) steps in the ith one, and $p_{ij}(x,y)$ is the probability assigned to the jth rule of the ith derivation.

Similar to the stochastic syntax analysis for context-free programmed languages [10], depending on the type of stochastic selection for the next rule, we have two distinct types of stochastic syntax analyzer for tree translation. The first type is one with a selection method which searches for the most likely rule first, while the second type is one with a selection method which randomly selects a rule for a nonterminal according to the distribution over all possible alternatives for the nonterminal considered.

The stochastic syntax analyzer that searches for the most likely rule first is a nondeterministic syntax analyzer in which the rules are arranged in descending order of magnitude of their associated probabilities [54]. The first rule for each nonterminal is the one with the highest probability, while the last rule for each nonterminal is the one corresponding to the lowest probability. Clearly, this is a stochastic syntax analyzer with a fixed strategy [10]. The procedure for this analyzer is given in Algorithm 3.2. A flow chart is shown in Fig. 3.2.

The stochastic syntax analyzer with a random strategy is formed from the nondeterministic syntax analyzer by incorporating a stochastic selection algorithm in selecting the next rule when alternatives are available. At each step the conditional probability distribution is used in selecting the next rule among all available alternatives [10]. Suppose that there are 3 translation rules for nonterminal A, γ_1, γ_2 and γ_3 with P_1, P_2, P_3 being their respective probabilities, and γ_1 has been previously applied. After the backtracking, γ_2 and γ_3 are the only available rules. Hence, either γ_2 or γ_3 can be selected according to their conditional probability distribution $P(\gamma_2,\gamma_3|\gamma_1) = (P_2',P_3')$ where $P_2' = P_2/(P_2+P_3)$ and $P_3' = P_3/(P_2+P_3)$. The procedure for this analyzer is given in Algorithm 3.3 and a flow chart is shown in Fig. 3.3. (Algorithm 3.2 of the SGSDTT parser with a fixed strategy is essentially the same as Algorithm 3.1 of the nonstochastic GSDTT parser. Algorithm 3.3 of the SGSDTT parser with a random strategy is almost the same as Algorithm 3.1 except the rule selection method.)

<u>Algorithm</u> <u>3.2</u>: Parsing for SGSDTT with a fixed strategy.

Input. An SGSDTT $F=(G, \Delta, \Gamma, R)$, $G=(V, \Sigma, P, S)$ and an input tree pair $<r,t>$.

Output. All the correct parses for $<r,t>$. The output "error" if no parse exists.

Method.

(1) For each nonterminal A in V, order the alternates in R for A in descending order of magnitude of their associated probabilities. Let A_{io} be the index for the ith alternate of A. Let A_{ij} be the index for the jth alternate in R with respect to A_{io} in P.

(2) Let $<u,v>$ be a new tree pair. A 4-tuple (X, b, l, k) will be used to denote the configuration of each node in a tree:

(a) X denotes the nonterminal of the node before it is rewritten.

(b) b denotes the terminal of the node after it is rewritten.

(c) l denotes the label of the production rule for rewriting the node.

(d) k denotes the sequence number for counting the sequence of rule applications.

(3) let $X = S$, $i=1$, $\langle u,v \rangle = \langle (S, \ , 0, 1), (S_1, \ , 0, 1) \rangle$.

(4) Select the first rule of P for X. j=1.

(5) replace node $(X, \ , l, k)$ of u with rule X_{jo} in P. If X_{jo} is $X \rightarrow b[B_1, B_2, \ldots, B_m]$, then replace the node with $(X, b, j, i)[(B_1, \ , 0, i), \ldots, (B_m, \ , 0, i)]$.

(6) Check compatibility for u and r, if compatible go to (7), else go to (11).

(7) For each node $(X_n, \ , 0, k)$ of v, where n is any number, replace it with rule X_{jn} in R. If a node in X_{jn} is a terminal b, it is written as (X_n, b, j, i), if a node in X_{jn} is a nonterminal B_m, it is written as $(B_m, \ , 0, i)$, go to (8).

(8) Check compatibility for v and t, if compatible, go to (9) else (12).

(9) Check if there is any nonterminal node left in u, if yes, let X be the next nonterminal, i=i+1, go to (4), if not, go to (10).

(10) Check compatibility for u and r, v and t, if compatible, find the parse by tracing down tree u, report success and go to (12), if not, go to (12).

(11) delete the subtree just being added to u, restore its root with $(X, \ , j, k)$. go to (13).

(12) delete the subtree just being added to u, restore its root with

(X, , j, k), delete the subtrees being added to v, restore their roots with (X_n, , j, k) go to (13).

(13) check if there are new rules available for X, if yes, set j=j+1. go to (5), if not go to (14).

(14) If X=S, then finished, exit, if not, then backtrack, take the immediate father of node (X, , j, k) as new X, i=i-1, and go to (4) (Figure 3.2 shows a flow chart for this algorithm).

Algorithm 3.3: Parsing for SGSDTT with a random strategy.

Input. An SGSDTT F=(G, Δ, Γ, R), G=(V, Σ, P, S) and an input tree pair <r,t>.

Output. All the correct parses for <r,t>. The output "error" if no parse exists.

Method.

(1) For each nonterminal A in V, order the alternates in R for A. Let A_{io} be the index for the ith alternate of A. Let A_{ij} be the index for the jth alternate in R with respect to A_{io} in P.

(2) Let <u,v> be a new tree pair. A 4-tuple (X, b, ℓ, k) will be used to denote the configuration of each node in a tree:

(a) X denotes the nonterminal of the node before it is rewritten.

(b) b denotes the terminal of the node after it is rewritten.

(c) ℓ is a register denoting which rules have been applied for the node before.

(d) k denotes the sequence number for counting the sequence of rule applications.

(3) let X = S, i=1, <u,v> = <(S, , 0, 1), (S_1, , 0, 1) >.

(4) Select the first rule of P for X statistically, set j = the index

of the rule.

(5) replace node (X, , l, k) of u with rule X_{jo} in P. If X_{jo} is $X \rightarrow b[B_1, B_2, \ldots, B_m]$, then replace the node with (X, b, ℓ', i)[(B_1, , 0, i), ..., (B_m, , 0, i)], ℓ' = (ℓ OR γ), where $\gamma = 2^j$, the jth bit of the binary form of γ is 1, showing that the jth rule of X is now being applied.

(6) Check compatibility for u and r, if compatible go to (7) else go to (11).

(7) For each node (X_n, , 0, k) of V, where n is any number, replace it with rule X_{jn} in R. If a node in X_{jn} is a terminal b, it is written as (X_n, b, ℓ', i), if a node in X_{jn} is a nonterminal B_m, it is written as (B_m, , 0, i), go to (8).

(8) Check compatibility for v and t, if compatible, go to (9) else (12).

(9) Check if there is any nonterminal node left in u, if yes, let X be the next nonterminal, i=i+1, go to (4), if not, go to (10).

(10) Check compatibility for u and r, v and t, if compatible, find the parse by tracing down tree u, report success and go to (12), if not, go to (12).

(11) delete the subtree just being added to u, restore its root with (X, , j, k). go to (13).

(12) delete the subtree just being added to u, restore its root with (X, , j, k), delete the subtrees being added to v, restore their roots with (X_n, , j, k) go to (13).

(13) check if there are new rules available for X, if yes, select next rule for X statistically, set j = the index of the rule, go to (5), if

not go to (14).

(14) If X=S, then finished, exit. If not, then backtrack, take the immediate father of node (X, , j, k) as new X, i=i-1, and go to (4) (Figure 3.3 shows a flow chart for this algorithm).

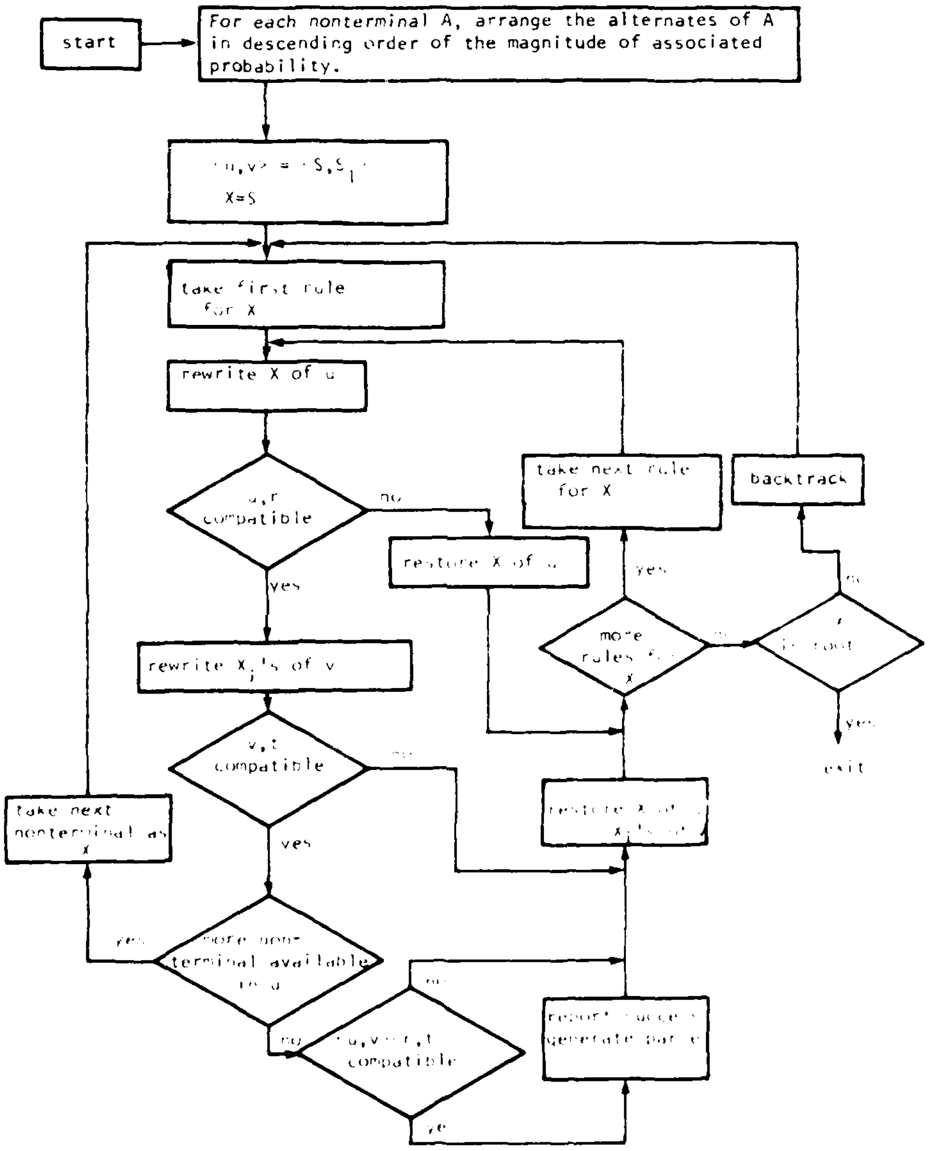


Figure 3.2 Flow chart for the parsing of an SGSDTT with a fixed strategy

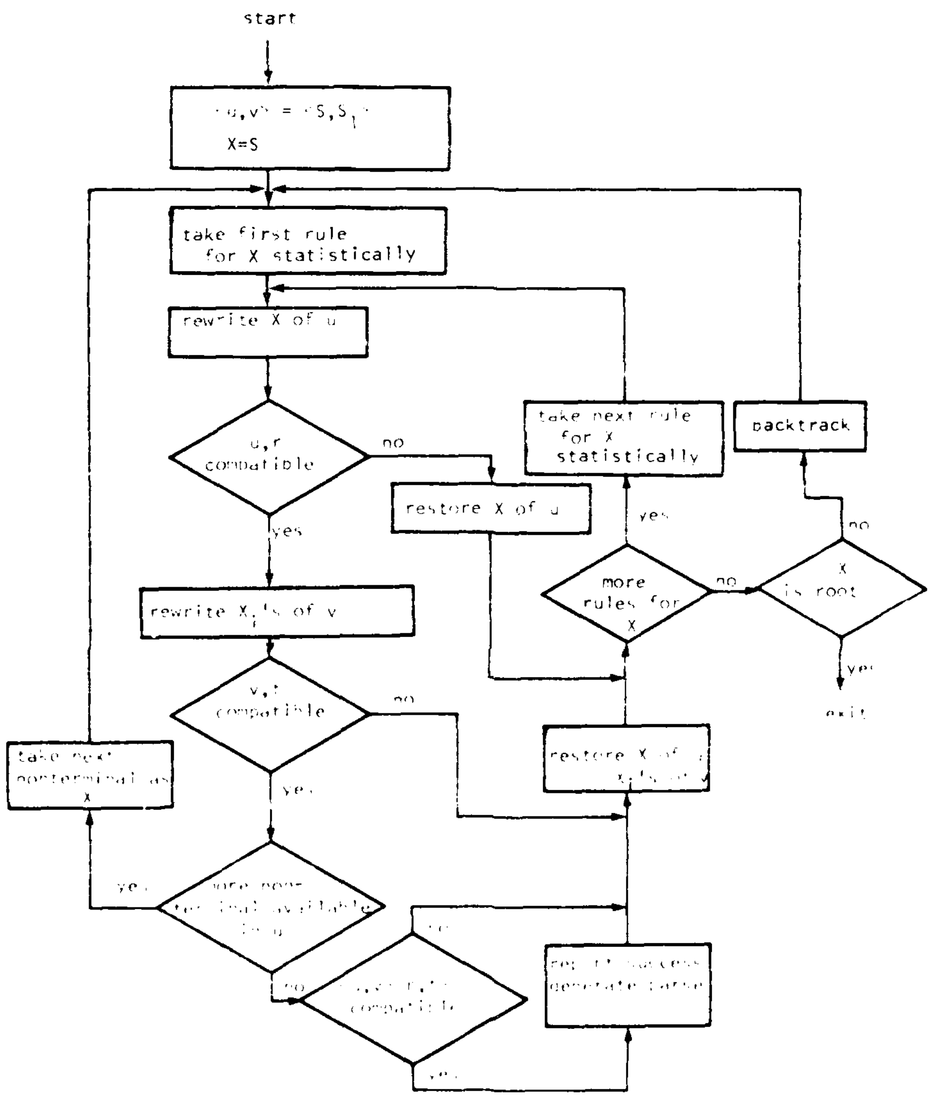

start
<u,v> = <S,S_1>
X=S
take first rule for X statistically
rewrite X of u
u,r compatible
no
yes
restore X of u
take next rule for X statistically
backtrack
yes
more rules for X
no
X is root
no
yes
exit
rewrite X_i's of v
v,t compatible
no
yes
restore X of u X_i's of v
take next nonterminal as X
yes
more non-terminal available in u
no
yes
rep. if success generate parse

Figure 3.3 Flow chart for the parsing of an SGSDTT with a random strategy

CHAPTER 4

AN APPLICATION OF TREE TRANSLATION TO TRAFFIC IMAGE SEQUENCE ANALYSIS

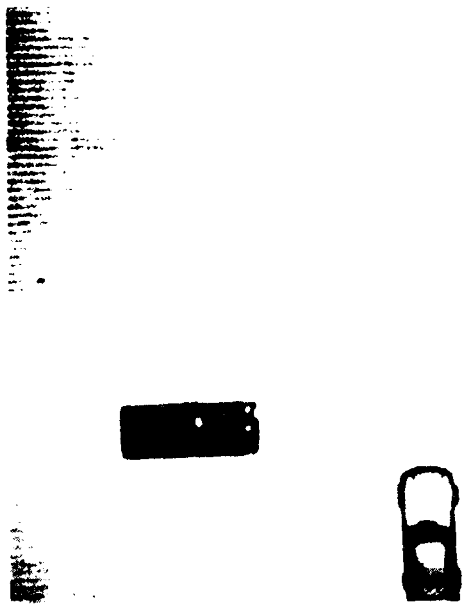
4.1 Introduction

This chapter gives an illustrative example of applying the tree translation model described in Chapter 3 to the design of an automatic traffic image sequence analysis system. Traffic image is a popular subject of study in time-varying image analysis. The input of the analysis system is assumed to be a sequence of images of a traffic intersection scene. An example is shown in Figure 4.1. In this system, each input image is divided into a set of fixed-size, fixed-position windows. Each image is then represented as a tree of which each node corresponds to a specific window in the image. Each node is labeled to indicate the occurrence or nonoccurrence of a vehicle in its corresponding window. A tree translation scheme is then used to describe the motion of the vehicles in the image sequences. Matching of vehicles in different images is performed in the form of a tree translation parsing.

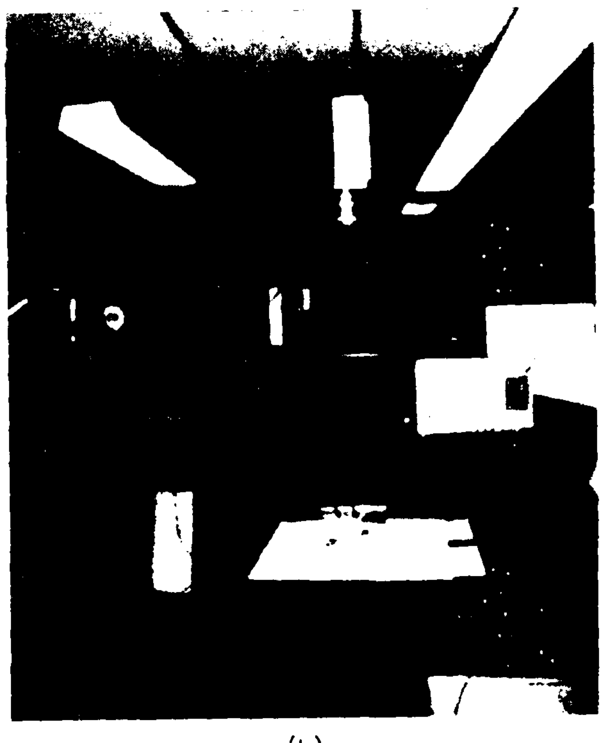
4.2 Scene Representation

4.2.1 Image representation

Each image is first divided into a set of windows. An appropriate window size is selected so that each window contains at most one



(a)



(b)

Figure 4.1 (a) Sample image (b) Data collection environment vehicle. In general, the window length selected should not be greater than the length of the smallest vehicle. Window size could be set at a very small value. However, a smaller window size will result in a more complex translation scheme. There are many ways to divide the image. An example is shown in Figure 4.2. Only a part of the road area is considered. A tree representation for the image is also shown in Figure 4.2. Each window corresponds to a node in the tree representation regardless there is a vehicle in this window or not. Each vehicle is considered as only a point. This point could be the centroid, or any easily-recognizable corner-point of the vehicle. But after this reference point is selected, this point should be used consistently for each vehicle in the whole image sequence. The centroid of a vehicle (considering only the x-y plane) is selected in this system. A vehicle's position is represented by the location of its centroid. Even though one large vehicle could occupy more than one window, only the window where this vehicle's centroid resides is considered as containing this vehicle. If no vehicle occurs in a window, then the corresponding node of this window is labeled by '0'. If there is a vehicle in a window, then the corresponding node is labeled by the quantization value of this vehicle's orientation (see Figure 4.3). Additional information (includes vehicle size, actual centroid position, orientation angle) is attached to the node for other purposes (e.g. speed calculation, clearing of ambiguity occurring in tree translation parsing). The determination of a vehicle's orientation and centroid position will be discussed later. An ambiguity of tree translation parsing means that different time-varying activities between images result in the same

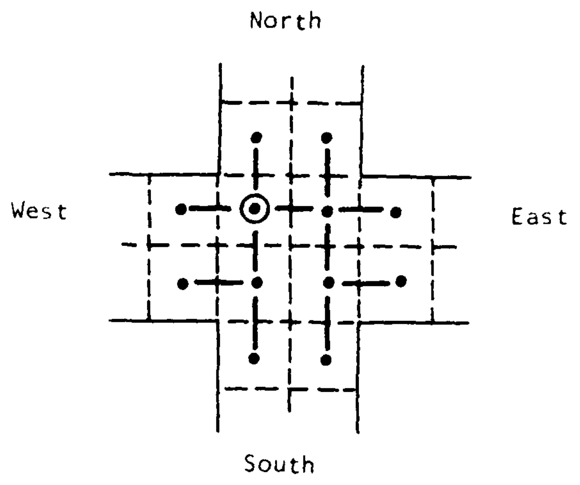


Figure 4.2 Divided intersection area and its tree representation. (The root of the tree is represented by an encircled node.)

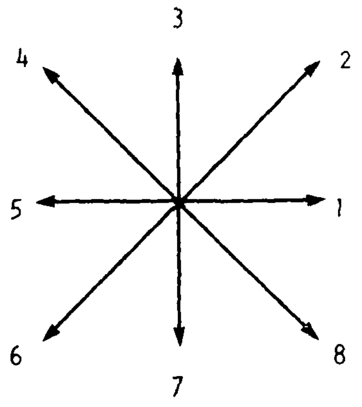


Figure 4.3 Orientation primitives

AD-A101 665   PURDUE UNIV LAFAYETTE IN SCHOOL OF ELECTRICAL ENGINEERING   F/G 9/2
A SYNTACTIC METHOD FOR TIME-VARYING PATTERN ANALYSIS.(U)
MAY 81   T J FAN, K S FU   N00014-79-C-0574
UNCLASSIFIED   TR-EE81-10   NL

2 OF 2
AD A
0101665

END
DATE
FILMED
8-81
DTIC

sequence of trees.

4.2.2 Motion representation

Tree translation rules are used to represent vehicle movement. Only the movement from one window to another is described. The movement within the same window is not considered. But the information about vehicle's centroid position and orientation is attached to the node. This information can be used if necessary. It is assumed that each vehicle can move no more than the length of the smallest vehicle between two consecutive images.

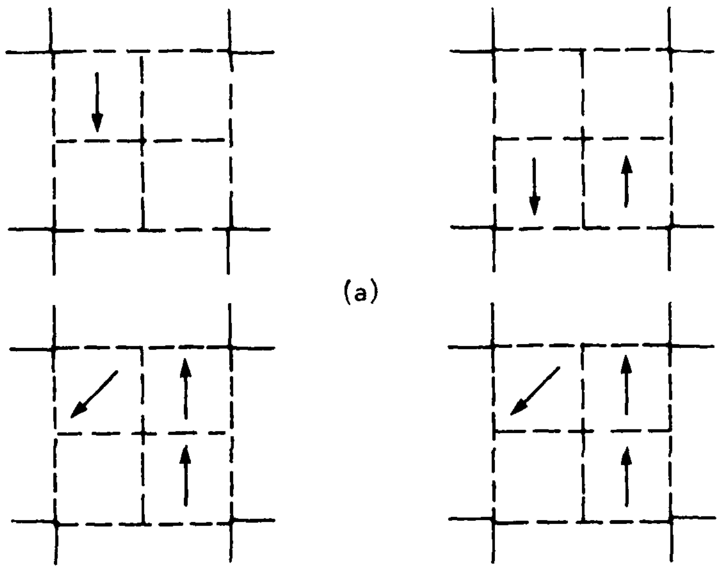
Example 4.1: Figure 4.4a shows an intersection area being divided into 4 windows. One vehicle is coming in from the south while another one is moving toward the south. The following rules are required for this movement:

```
S →  7      0             B → 0   0
    /\     /\                 |   |
   A  B , A  B                C , C


A →  0  ,   7             C → 0 , 3
```

where 0 means no vehicle existence and i means the existence of a vehicle in direction i, $1 \leq i \leq 8$.

Example 4.2: Figure 4.4b shows that one vehicle is moving from the north to the west and two others are moving from the south to the north. Both the (i-1)th and the ith images have the same intersection content. There are two interpretations for the vehicle moving west. Either the



(a)

(b)

Figure 4.4 Intersection area contents. ("↑" sign indicates the existence of a vehicle in that direction.)

same vehicle exists in the same window for both images or the one in the (i-1)th image has left the intersection area and a new one is appearing in the ith image. A similar interpretation also holds for those two vehicles moving north. There can be four different interpretations for the variation between the (i-1)th and the ith images. But only one set of translation rules is needed to describe this image pair.

$$\begin{tikzcd}[column sep=small] S \rightarrow & & 6 \arrow[dl, dash] \arrow[dr, dash] & & & 6 \arrow[dl, dash] \arrow[dr, dash] & \\ & A & & B \; , & A & & B \end{tikzcd}
\qquad
\begin{tikzcd}[column sep=small] B \rightarrow & 3 \arrow[d, dash] & & 3 \arrow[d, dash] \\ & C & , & C \end{tikzcd}$$

$$A \rightarrow 0 \; , \; 0 \qquad\qquad C \rightarrow 3 \; , \; 3$$

From the assumption about vehicle speed, this problem is solved by using vehicle position and orientation values attached to each node of the tree. For the vehicle with label 6, if the orientation value of the vehicle in the ith image is greater than that of the vehicle in the (i-1)th image, then it indicates that the vehicle in the (i-1)th image has left the intersection and the vehicle in the ith image appears for the first time and traffic flow (the number of passing vehicles) is increased by one. Otherwise it indicates that the vehicles appearing in the (i-1)th and the ith images are the same ones. (Orientation value is counted counterclockwise) For those vehicles moving north, the y coordinate of the front vehicle is used. If the y coordinate of the front vehicle in the ith image is smaller than that of the front vehicle in the (i-1)th image, then it indicates that the front vehicle in the (i-1)th image has left the intersection, the front vehicle in the ith

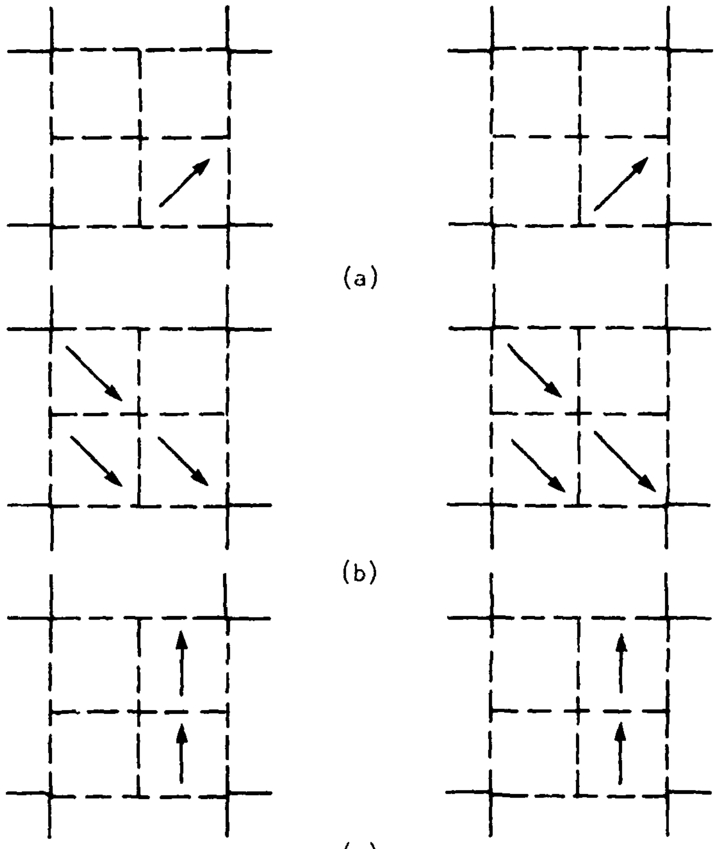


Figure 4.5 Conditions of the (i-1)th and ith images for which the use of attribute value is required. (Each one of (a) (b) (c) represents one of four similar cases. Blank windows could contain vehicle or not. The attribute values of node representing lower-right window will be used for case (a) and (b). For case (c) the attribute values of the top-right window will be used.)

image was the rear vehicle in the (i-1)th image, the rear vehicle in the ith image appears for the first time and the traffic flow is increased by one. Otherwise it indicates that the front vehicles in both the ith and the (i-1)th images are the same one and so are the rear vehicles. There are also some other similar cases which require the use of attribute values of each node to resolve such an ambiguity. Their corresponding translation rules are specified for the requirement of the comparison between node attribute values of 2 input trees. Figure 4.5 shows the various cases in which a comparison of node attribute value is required.

4.3 Scene Analysis

4.3.1 Feature extraction

Due to the similar property in feature extraction between this experiment and You and Fu's shape recognition experiment [13], the algorithms for boundary following and boundary smoothing described in [13] are applied for feature extraction.

The first step in feature extraction is threshold selection. Because of the flat black paint on the models, the vehicles look uniformly dark and should create a peak in the high gray level region of the histogram. The light background is supposedly uniform too and create a peak in the low gray level region of the histogram. A typical histogram of the experiment is shown in Figure 4.6. The peak at gray level k_1 is caused by the vehicles. The peak at gray level k_2 is caused by the background. The selected threshold is t.

<u>Algorithm 4.1</u>: Threshold Selection

Input: A digital picture.

Output: A threshold t.

Method:

(1) Compute the histogram.

(2) Find the peak of the highest gray level (which usually corresponds to the object). Let k_1 = the gray level.

(3) Find the second peak of the histogram besides the one found in (2). Let k_2 = the corresponding gray level.

(4) Find the lowest valley between the above two peaks and let t = gray level corresponding to the valley.

(5) Terminate.

After a threshold is found, the boundary for each vehicle can be traced out. The boundary is defined as a connection of edges between the object and the background. The boundary is coded by unit vectors with horizontal and vertical directions. Each boundary is traced out by the following boundary following algorithm which is led by the contents of a 2x2 window. Figure 4.7 shows the four possible configurations. The pixels A, B, C, and D are defined relative to the boundary vector P. The object is to the right of P, so that A is darker than the threshold t. The background is to the left of P, so that C is lighter than t. In the following algorithm, u_x and u_y are the unit movements, or unit vectors, in the X and Y directions respectively. A, B, C, D denote the coordinates of the pixels. G(B) is the gray level of the pixel indicated by B. F is the first pixel of an object detected by scanning the image.

<u>Algorithm 4.2</u>: Boundary Following (You and Fu [13])

Input: F, u_x, u_y, and threshold t.

Output: A boundary chain U of n unit vectors.

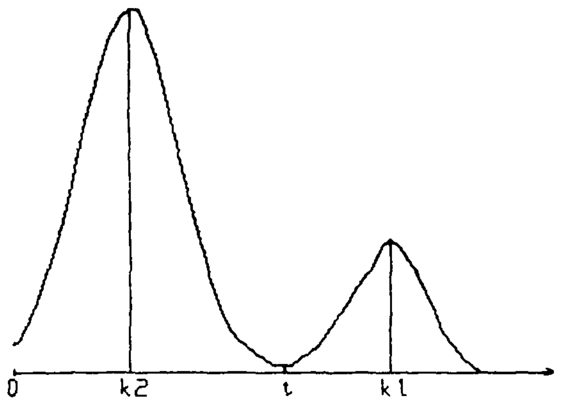


Figure 4.6 A typical histogram of the experiment

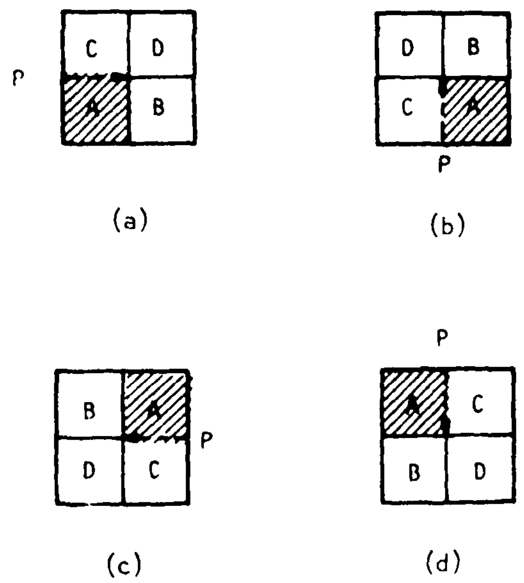


Figure 4.7 Four possible configurations of boundary following window

Method:

```
(1)  Set $P = u_x$, $i = 1$, $U(1) = P$
         $A = F$, $C = F-u_y$, $S = C$, go to (3)
(2)  If $(S = C)$ then terminate
     otherwise $i = i+1$, $U(i) = P$
(3)  $D = C+P$
     If $(G(D) < t)$ then go to (4)
     otherwise $P = C-A$, $A = D$, go to (2)
(4)  $B = A+P$
     If $(G(B) < t)$ then go to (5)
     otherwise $A = B$, $C = D$, go to (2)
(5)  $C = B$, $P = B-D$ go to (2)
```

The output from the above boundary following algorithm is a string of unit vectors. Due to the digitization grid, there are many zig-zag's. A smoothing method (You and Fu [13]) is applied to transform a string of unit vectors into a string of longer vectors. This method is defined in the form of an attributed finite transducer.

In the following definition, qj's are attributed states. Each state qj represents a subchain sj which is accepted by not translated, and which is described by the associated attributes. ℓv denotes a series of unit vector v, or ℓ times v. $-v$ is the negative of v, i.e., $-v$ and v have the same length but opposite directions. $A \wedge B$ denotes that B follows A. δ is a mapping from $Q \times I$, under condition C, to finite subsets of $Q \times O^*$. The mapping performs when condition C is true. For each state transition, there is a set of attribute rules.

The attributes of a state may be unit vectors or numbers. In the transition rule, i, m, n, v, u are unit vectors and p, q, ℓ, k are numbers. The attribute conditions for a transition are described above the right arrow. For each transition, the input unit vector is compared with the vector attributes and the attribute conditions are checked. Then the machine goes to the next state with the appropriate output and transfers the attributes according to the attribute rules. Each expression of the output is a vector $\in$ O. An example of the smoothing effect of this transducer is shown in Figure 4.8.

Definition 4.1: Attributed Finite Transducer A

A is a 6-tuple, (Q, I, O, δ, S, F).

I = the input set consisting of 4 unit vectors and an end marker \$, $\{(1,0),(0,-1),(-1,0),(0,1),\$\}$

O = the output set, $\{(n_1,n_2) \mid (n_i = 0, n_{3-i} \neq 0)$ or $(n_i = \pm 1,\ n_{3-i} = \text{any integer}),\ i = 1,2\}$

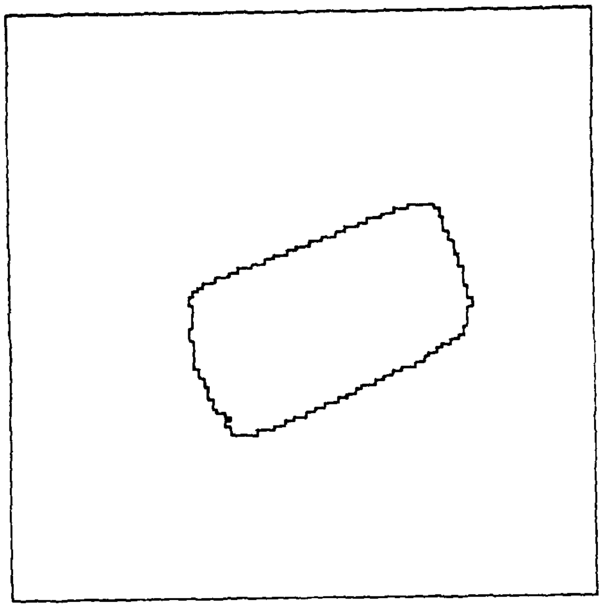
Q = a set of states with attributes, $\{qj \mid j = 0,\dots,9\}$

$q0$: $s0 = \lambda$, empty

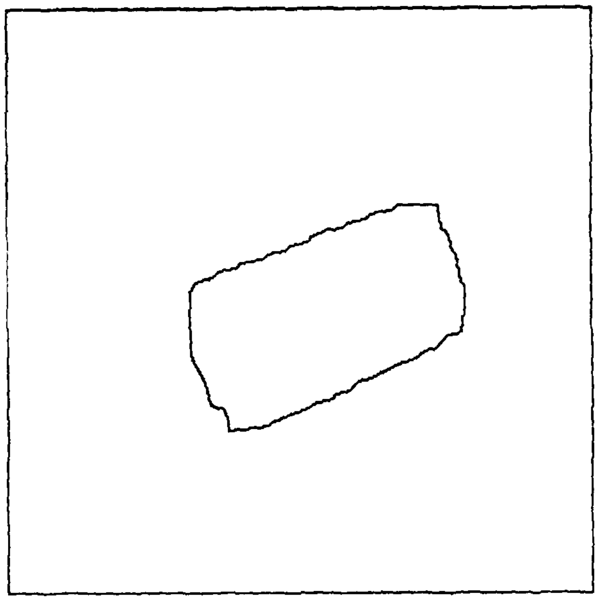
$q1_v$: s1 has one unit vector v

$q2_{\ell,v}$: $s2 = \ell v, \underset{\ell v}{\longrightarrow}, \ \ell > 1$

$q3_{v,u}$: $s3 = v \wedge u,\ \overset{v}{\urcorner\!\downarrow}\ u$ (or $\lrcorner\!\uparrow$)

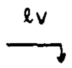


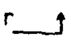
(a)

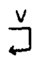


(b)

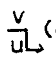
Figure 4.8 (a) Detected boundary (b) Smoothed boundary

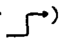
$q4_{\ell,v,u}$: $s4 = \ell v \wedge u$, 

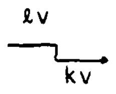
u (or ), $\ell > 1$

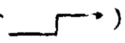
$q5_{v,u}$: $s5 = v \wedge u \wedge -v$, 

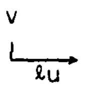
u (or)

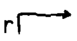
$q6_{v,u}$: $s6 = v \wedge u \wedge v$, 

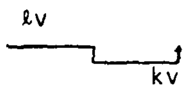
(or )

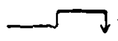
$q7_{\ell,v,u,k}$: $s7 = \ell v \wedge u \wedge kv$, 

(or )

$q8_{v,\ell,u}$: $s8 = v \wedge \ell u$, 

(or ), $\ell > 1$

$q9_{\ell,v,uk}$: $s9 = \ell v \wedge u \wedge kv \wedge -u$, 

(or ),
$\ell > 1$, $k > 1$

S = the initial state q0

F = a set of final states, {q0}

δ: $(q0,i) \rightarrow (q1_v,\phi)$, $v \leftarrow i$, $i \in I-\{\$\}$

$(q1_m,m) \rightarrow (q2_{\ell,v},\phi)$, $v \leftarrow m$, $\ell \leftarrow 2$

$(q1_m,i) \rightarrow (q3_{v,u},\phi)$, $v \leftarrow m$, $u \leftarrow i$

$(q2_{p,m}, m) \rightarrow (q2_{\ell,v}, \phi),\ v \leftarrow m,\ \ell \leftarrow p+1$

$(q2_{p,m}, i) \rightarrow (q4_{\ell,v,u}, \phi),\ \ell \leftarrow p,\ v \leftarrow m,\ u \leftarrow i$

$(q3_{m,n}, m) \rightarrow (q6_{v,u}, \phi),\ v \leftarrow m,\ u \leftarrow n$

$(q3_{m,n}, n) \rightarrow (q8_{v,\ell,u}, \phi),\ v \leftarrow m,\ u \leftarrow n,\ \ell \leftarrow 2$

$(q3_{m,n}, -m) \rightarrow (q5_{v,u}, \phi),\ v \leftarrow m,\ u \leftarrow n$

$(q4_{p,m,n}, m) \rightarrow (q7_{\ell,v,u,k}, \phi),\ \begin{matrix} \ell \leftarrow p,\ v \leftarrow m \\ k \leftarrow 1,\ u \leftarrow n \end{matrix}$

$(q4_{p,m,n}, -m) \rightarrow (q3_{v,u}, pm),\ v \leftarrow n,\ u \leftarrow v$

$(q4_{p,m,n}, n) \rightarrow (q2_{\ell,v}, pm),\ \ell \leftarrow 2\ v \leftarrow n$

$(q5_{m,n}, -n) \rightarrow (q0, \phi)$

$(q5_{m,n}, -m) \rightarrow (q2_{\ell,v}, m+n),\ \ell \leftarrow 2,\ v \leftarrow -m$

$(q5_{m,n}, n) \rightarrow (q2_{\ell,v}, \phi),\ \ell \leftarrow 2,\ v \leftarrow n$

$(q6_{m,n}, -n) \rightarrow (q9_{\ell,v,u,k}, \phi),\ \begin{matrix} v \leftarrow m,\ \ell \leftarrow 1 \\ u \leftarrow n,\ k \leftarrow 1 \end{matrix}$

$(q6_{m,n}, n) \rightarrow (q0, m+n \,/\, m+n)$

$$(q6_{m,n}, m) \rightarrow (q7_{\ell,v,u,k}, \phi), \begin{array}{l} \ell \leftarrow 1,\ k \leftarrow 2 \\ v \leftarrow m,\ u \leftarrow n \end{array}$$

$$(q7_{p,m,n,q}, m) \rightarrow (q7_{\ell,v,u,k}, \phi), \begin{array}{l} v \leftarrow m,\ \ell \leftarrow p \\ u \leftarrow n,\ k \leftarrow q+1 \end{array}$$

$(q7_{p,m,n,q}, n)$

$$\xrightarrow{p>2q} (q1_v, (p-1)m/\ 2qm+n),\ v \leftarrow n$$

$$\xrightarrow{q/2<p<2q} (q1_v, (p+q)m+n),\ v \leftarrow n$$

$$\xrightarrow{p<q/2} (q4_{\ell,v,u}, 2pm+n), \begin{array}{l} \ell \leftarrow q-p \\ v \leftarrow m \\ u \leftarrow n \end{array}$$

$$(q7_{p,m,n,q}, -n) \rightarrow (q9_{\ell,v,u,k}, \phi) \begin{array}{l} \ell \leftarrow p \\ k \leftarrow q \\ v \leftarrow m \\ u \leftarrow n \end{array}$$

$$(q8_{m,q,n}, n) \rightarrow (q8_{v,\ell,u}, \phi), \begin{array}{l} v \leftarrow m \\ u \leftarrow n \\ \ell \leftarrow q+1 \end{array}$$

$(q8_{m,q,n}, m)$

$$\xrightarrow{q\leq 4} (q4_{\ell,v,u}, m+n) \begin{array}{l} \ell \leftarrow q-1 \\ v \leftarrow n \\ u \leftarrow m \end{array}$$

$$\xrightarrow{q>4} (q4_{\ell,v,u}, m+2n) \begin{array}{l} \ell \leftarrow q-2 \\ v \leftarrow n \\ u \leftarrow m \end{array}$$

$(q8_{m,q,n}, -m)$

$$\xrightarrow{q\leq 4} (q4_{\ell,v,u}, m+n) \begin{array}{l} \ell \leftarrow q-1 \\ v \leftarrow n \\ u \leftarrow -m \end{array}$$

$$\xrightarrow{q>4} (q4_{\ell,u,v}, m+2n) \begin{array}{l} \ell \leftarrow q-2 \\ v \leftarrow n \\ u \leftarrow -m \end{array}$$

$$(q9_{p,m,n,q}, m) \to (q2_{\ell,v}, \phi) \begin{array}{l} \ell \leftarrow m+n+1 \\ v \leftarrow m \end{array}$$

$$(q9_{p,m,n,q}, -n) \to (q2_{\ell,v}, (p+q)m+n), \begin{array}{l} \ell \leftarrow 2 \\ v \leftarrow -n \end{array}$$

$$(q9_{p,m,n,q}, -m) \to (q3_{v,u}, (p+q)m+n), \begin{array}{l} v \leftarrow -n \\ u \leftarrow -m \end{array}$$

$(q0,\$) \to (q0,\phi)$

$(q1_m,\$) \to (q0,m)$

$(q2_{p,m},\$) \rightarrow (q0,pm)$

$(q3_{m,n},\$) \rightarrow (q0,m+n)$

$(q4_{p,m,n},\$) \rightarrow (q0,pm+n)$

$(q5_{m,n},\$) \rightarrow (q0,n)$

$(q6_{m,n},\$) \rightarrow (q0,2m+n)$

$(q7_{p,m,n,q},\$) \rightarrow (q0,(p+q)m+n)$

$(q8_{m,q,n},\$) \rightarrow (q0,m+qn)$

$(q9_{p,m,n,q},\$) \rightarrow (q0,(p+q)m)$

4.3.2 Shape analysis

Consider the top view first. An attributed shape grammar is adapted for vehicle shape representation [13]. To find the orientation and the centroid position of a vehicle, special symbols are marked in the production rules of the shape grammar to specify which pair of angle points can be used to find the orientation and which pair of angle points can be used to find the centroid position. Suppose that a derivation of a shape contour is $S \overset{*}{\rightarrow} F_1A_1 \ldots F_iA_i \ldots F_jA_j \ldots A_m$, where F's are curve primitives and A's are angle primitives, and it is known in the grammar inference stage that the direction of the line segment extending from the point of angle A_i to the point of angle A_j decides the orientation of this vehicle and it is also known from parsing that this derivation generates the input boundary string $V_1 \ldots V_p \ldots V_q \ldots V_n$ with substring $V_p V_{p+1} \ldots V_q$ matching $F_{i+1} \ldots F_j$. Then the position of angle A_i (A_j) is the same as the position of the breaking point between vectors V_{p-1} and V_p (V_q and V_{q+1}). Therefore the orientation of this vehicle is decided with the knowledge of x and y coordinates of these two breaking points which is obtained in the feature extraction stage. The centroid position is calculated similarly. Suppose that the midpoint of the line segment connecting the angles A_k and A_l is the centroid. Then the centroid position is decided from the x and y coordinates of the angle points of A_k and A_l. You and Fu's [13] PEE Earley's parser is used for vehicle shape recognition.

For example, consider a vehicle shape shown in Figure 4.9. The shape is represented as $S \rightarrow F_1A_1F_2A_2F_3A_3F_4A_4$. The midpoint of the line segment connecting the angles A_2 and A_4 is the centroid. The direction of the line segment connecting the angles A_1 and A_2 determines the orientation of the vehicle. After knowing that A_1 is located at (x_1,y_1), A_2 is located at (x_2,y_2) and A_3 is located at (x_3,y_3), then the centroid position is found to be $((x_2+x_4)/2, (y_2+y_4)/2)$ and the orientation of the vehicle has the slope $(y_2-y_1)/(x_2-x_1)$. The question about whether the actual direction should be from A_1 to A_2 or from A_2 to A_1 will be discussed shortly. Strictly speaking, for a simple shape like the top view of a vehicle, the shape analysis method described above is not really necessary. For instance, an easy way to calculate the centroid (x_c,y_c) of a vehicle is to apply the formula: $x_c=\sum_i x_i/n$,

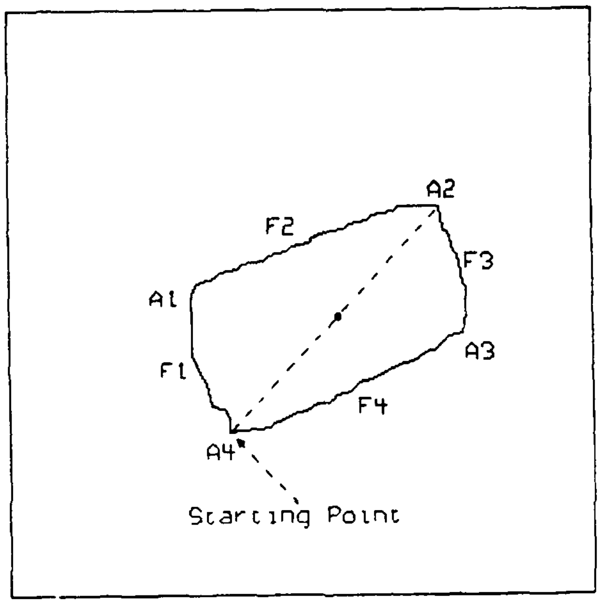


Figure 4.9 An example of vehicle shape

$y_c=\sum_i y_i/n$, where the summation is taken over all points on the boundary and n is the total points on the boundary. The orientation can also be easily determined after searching for the right angles around the boundary and finding out the longest line segment between these angles. But compared with this simple method, the method described earlier in this subsection does serve as a way for the determination of the centroid and orientation of a general shape.

Due to the symmetrical property of the top view of a vehicle, a vehicle in any direction could have two possible opposite orientations. For example, the shape of a vehicle moving northwest could also be evaluated as southeast - oriented. This problem is solved as follows: Consider a vehicle located in the lower-right window of the intersection area. If its shape is detected as horizontally oriented (either to the east or west), then it is considered as moving east with orientation 1. Similarly, if its shape is detected as vertically oriented, then it is considered as moving north with orientation 3. If its shape is northwest - oriented (in other terms, southeast - oriented), it could have orientation 4 or 8. this ambiguity is resolved from the knowledge of the last image and the assumption about vehicle movement limitation between consecutive images. Specifically, if there exists a vehicle with orientation 8 in either the lower-left or the lower-right window of the intersection area in the (i-1)th image, then this northwest-oriented vehicle in the lower-right window of the ith image should have orientation 8 instead of 4. Otherwise this vehicle should have orientation 4. For a northeast-oriented vehicle in the lower-right window, there is only one choice: orientation 2. The way of determining the orientation of vehicles in the lower-right window described above also applies to other windows.

When an input image does not represent the top view, the vehicle shape variation and occlusion problems happen. For shape variation and simple occlusion, the problem can still be solved with the help of a generalized error-correcting PEE parser (GECPEEP) [13]. But when the number of vehicles appearing in the scene increases, a serious occlusion would make the extraction of vehicle centroid position and orientation information very difficult. Therefore, in order to make the monitor working even under heavy traffic conditions, it would be better to use the top-view of the scene as the input.

4.3.3 Tree translation parsing

After the tree representation for the ith image is obtained, it is parsed by the tree translation parser along with the tree representation for the (i-1)th image. (Parsing process starts from the second image.) The translation parser is similar to the one shown in Chapter 3. But as mentioned before, some configurations of input tree pair still require the use of attributes attached to the node to decide the actual activity between two consecutive images. After the parsing, the parser either reports the output or asks for a comparison of attribute values. (The conditions requiring the use of attributes and which attributes to be used have been discussed earlier and illustrated in Figure 4.5.)

4.4 Inference of a Structure-Preserved Tree Translation Schema

The tree translation involved here is structure-preserved (in other words, substitution transformation only). The inference of the tree translation schema is therefore essentially a tree grammar inference problem: Let Σ be the input alphabet, Δ be the output alphabet of a tree translation and $S^+ = \{<x_1,y_1>, <x_2,y_2>\ldots\}$ be the tree translation sample set. x_i and y_i have the same tree structure. Form a new alphabet $\Sigma' = \Sigma \times \Delta$. Transform each tree pair $<x_i,y_i>$ into a single tree over Σ' by combining the label of each node of x_i and the label of its corresponding node of y_i. The tree translation sample set S^+ is now transformed into a set of tree samples. Apply the conventional tree grammar inference procedure (Moayer and Fu [61]) to infer the tree grammar which can generate these tree samples. Then transform this tree grammar into the required tree translation schema by converting each production rule into a tree translation rule. (e.g. if $A \rightarrow \overset{ab}{\underset{B_1 \quad B_n}{\diagup \ldots \diagdown}}$ is a rule of the tree grammar, where $a \in \Sigma$ and $b \in \Delta$, then the converted tree translation rule is $A \rightarrow \overset{a}{\underset{B_1 \quad B_n}{\diagup \ldots \diagdown}} \; , \; \overset{b}{\underset{B_1 \quad B_n}{\diagup \ldots \diagdown}}$). The whole procedure is given as follows:

<u>Algorithm 4.3</u>: Inference of a structured-preserved tree translation schema

Input: A tree translation sample set $S^+ = \{<x_1,y_1>, x_2,y_2>\ldots<x_n,y_n>\}$, where x_i and y_i have the same tree structure (tree domain), input alphabet Σ and output alphabet Δ.

Output: a tree translation schema which generates S^+.

Method: 1. Let $\Sigma' = \Sigma \times \Delta$, $S^{++} = \phi$.

2. For each $<x_i,y_i> \in S^+$,

add a tree $(\alpha\colon\ D \rightarrow \Sigma')$ to S^{++},

where D is the domain of x_i and y_i,

$\alpha(b) = \alpha_{xi}(b)\alpha_{yi}(b)$ for $b \in D$,

α_{xi} and α_{yi} are the tree function for x_i and y_i respectively.

3. Infer a tree grammar for S^{++} (Moayer and Fu [61]):

Step 1: Selection of proper substructures, here the aim is to select components which have high-repetitive occurrence.

Step 2: Infer subgrammar G_i for each substructure

Step 3: Let G_t be the union of the subgrammars G_i inferred in step 2. Relabel the variables and eliminate redundant production rules to eliminate ambiguity and erroneous generation paths.

4. Form a tree translation schema $T(\Sigma,\Delta,\gamma,R,S)$ by adding one rule $A \rightarrow \begin{array}{c} a \\ /\ldots\backslash \\ B_1 \quad B_n \end{array}, \begin{array}{c} b \\ /\ldots\backslash \\ B_1 \quad B_n \end{array}$ to R for each rule $A \rightarrow \begin{array}{c} ab \\ /\ldots\backslash \\ B_1 \quad B_n \end{array}$ of the grammar generated in 3.

Consider the tree translation involved in the traffic scene as an example. The input alphabet and output alphabet are $\Sigma = \Delta = \{0,1,2,3,4,5,6,7,8\}$. To reduce the number of samples, consider only the image pairs in which no vehicle exists in the intersection area of the first image and one vehicle comes into the intersection area of the second image. The tree translation sample set contains:

$$\left\langle \begin{array}{ccc} 0 & — & 0 \\ | & & | \\ 0 & & 0 \end{array}, \begin{array}{ccc} 0 & — & 0 \\ | & & | \\ 0 & & 2 \end{array} \right\rangle, \left\langle \begin{array}{ccc} 0 & — & 0 \\ | & & | \\ 0 & & 0 \end{array}, \begin{array}{ccc} 0 & — & 0 \\ | & & | \\ 0 & & 3 \end{array} \right\rangle,$$

```
   0—0   0—0       0—0   0—4
<  | | , | |  > , < | | , | |  > ,
   0 0   0 4       0 0   0 0


   0—0   0—5       0—0   0—6
<  | | , | |  > , < | | , | |  > ,
   0 0   0 0       0 0   0 0


   0—0   6—0       0—0   7—0
<  | | , | |  > , < | | , | |  > ,
   0 0   0 0       0 0   0 3


   0—0   8—0       0—0   0—0
<  | | , | |  > , < | | , | |  > ,
   0 0   0 0       0 0   8 0


   0—0   0—0       0—0   0—0
<  | | , | |  > , < | | , | |  > .
   0 0   1 0       0 0   2 0
```

These tree translation samples are transformed into trees over $\sum \times \Delta$.

Then a tree grammar is generated from these tree samples. For example,

```
  0—0   0—0            00—00
< | | , | | > becomes  |    |.  The tree sample set contains:
  0 0   0 2            00   02
```

```
00—00   00—00   00—00
|    | , |    |, |    | ,
00   02  00   03 00   04


00—04   00—05   00—06
|    | , |    |, |    | ,
00   00  00   00 00   00
```

$$\begin{tikzcd} 06 \arrow[r, dash] \arrow[d, dash] & 00 \arrow[d, dash] \\ 00 & 00 \end{tikzcd} , \quad \begin{tikzcd} 07 \arrow[r, dash] \arrow[d, dash] & 00 \arrow[d, dash] \\ 00 & 00 \end{tikzcd} , \quad \begin{tikzcd} 08 \arrow[r, dash] \arrow[d, dash] & 00 \arrow[d, dash] \\ 00 & 00 \end{tikzcd} ,$$

$$\begin{tikzcd} 00 \arrow[r, dash] \arrow[d, dash] & 00 \arrow[d, dash] \\ 08 & 00 \end{tikzcd} , \quad \begin{tikzcd} 00 \arrow[r, dash] \arrow[d, dash] & 00 \arrow[d, dash] \\ 01 & 00 \end{tikzcd} , \quad \begin{tikzcd} 00 \arrow[r, dash] \arrow[d, dash] & 00 \arrow[d, dash] \\ 02 & 00 \end{tikzcd} ,$$

The resulting tree grammar contains the following rules:

$$S \rightarrow \begin{tikzcd}[column sep=tiny] & 00 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_1 \end{tikzcd} \qquad S \rightarrow \begin{tikzcd}[column sep=tiny] & 00 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_2 \end{tikzcd}$$

$$S \rightarrow \begin{tikzcd}[column sep=tiny] & 00 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_3 \end{tikzcd} \qquad S \rightarrow \begin{tikzcd}[column sep=tiny] & 00 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_4 \end{tikzcd}$$

$$S \rightarrow \begin{tikzcd}[column sep=tiny] & 00 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_5 \end{tikzcd} \qquad S \rightarrow \begin{tikzcd}[column sep=tiny] & 00 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_6 \end{tikzcd}$$

$$S \rightarrow \begin{tikzcd}[column sep=tiny] & 06 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_7 \end{tikzcd} \qquad S \rightarrow \begin{tikzcd}[column sep=tiny] & 07 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_7 \end{tikzcd}$$

$$S \rightarrow \begin{tikzcd}[column sep=tiny] & 08 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_7 \end{tikzcd} \qquad S \rightarrow \begin{tikzcd}[column sep=tiny] & 00 \arrow[dl, dash] \arrow[dr, dash] & \\ A_2 & & B_7 \end{tikzcd}$$

$$S \rightarrow \begin{tikzcd}[column sep=tiny] & 00 \arrow[dl, dash] \arrow[dr, dash] & \\ A_3 & & B_7 \end{tikzcd} \qquad S \rightarrow \begin{tikzcd}[column sep=tiny] & 00 \arrow[dl, dash] \arrow[dr, dash] & \\ A_4 & & B_7 \end{tikzcd}$$

$$B_1 \rightarrow \begin{array}{c} 00 \\ | \\ A_4 \end{array} \qquad B_2 \rightarrow \begin{array}{c} 00 \\ | \\ C_1 \end{array}$$

$$B_3 \rightarrow \begin{array}{c} 00 \\ | \\ C_2 \end{array} \qquad B_4 \rightarrow \begin{array}{c} 04 \\ | \\ A_1 \end{array}$$

$$B_5 \rightarrow \begin{array}{c} 05 \\ | \\ A_1 \end{array} \qquad B_6 \rightarrow \begin{array}{c} 06 \\ | \\ A_1 \end{array}$$

$$B_7 \rightarrow \begin{array}{c} 00 \\ | \\ A_1 \end{array}$$

$$A_1 \rightarrow 00 \qquad A_2 \rightarrow 08$$

$$A_3 \rightarrow 01 \qquad A_4 \rightarrow 02$$

$$C_1 \rightarrow 03 \qquad C_2 \rightarrow 04$$

where S, $A_1,\ldots,A_4$, $B_1,\ldots,B_7$, C_1 and C_2 are nonterminals.

The resulting tree translation schema contains the following translation rules:

$$S \rightarrow \begin{array}{ccc} & 0 & \\ / & & \backslash \\ A_1 & & B_1 \end{array} , \begin{array}{ccc} & 0 & \\ / & & \backslash \\ A_1 & & B_1 \end{array} \qquad S \rightarrow \begin{array}{ccc} & 0 & \\ / & & \backslash \\ A_1 & & B_2 \end{array} , \begin{array}{ccc} & 0 & \\ / & & \backslash \\ A_1 & & B_2 \end{array}$$

$$S \to \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_3 \end{tikzcd} , \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_3 \end{tikzcd} \qquad S \to \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_4 \end{tikzcd} , \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_4 \end{tikzcd}$$

$$S \to \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_5 \end{tikzcd} , \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_5 \end{tikzcd} \qquad S \to \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_6 \end{tikzcd} , \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_6 \end{tikzcd}$$

$$S \to \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_7 \end{tikzcd} , \begin{tikzcd} & 6 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_7 \end{tikzcd} \qquad S \to \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_7 \end{tikzcd} , \begin{tikzcd} & 7 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_7 \end{tikzcd}$$

$$S \to \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_7 \end{tikzcd} , \begin{tikzcd} & 8 \arrow[dl, dash] \arrow[dr, dash] & \\ A_1 & & B_7 \end{tikzcd} \qquad S \to \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_2 & & B_7 \end{tikzcd} , \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_2 & & B_7 \end{tikzcd}$$

$$S \to \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_3 & & B_7 \end{tikzcd} , \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_3 & & B_7 \end{tikzcd} \qquad S \to \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_4 & & B_7 \end{tikzcd} , \begin{tikzcd} & 0 \arrow[dl, dash] \arrow[dr, dash] & \\ A_4 & & B_7 \end{tikzcd}$$

$$B_1 \to \begin{tikzcd} 0 \arrow[d, dash] \\ A_4 \end{tikzcd} , \begin{tikzcd} 0 \arrow[d, dash] \\ A_4 \end{tikzcd} \qquad B_2 \to \begin{tikzcd} 0 \arrow[d, dash] \\ C_1 \end{tikzcd} , \begin{tikzcd} 0 \arrow[d, dash] \\ C_1 \end{tikzcd}$$

$$B_3 \to \begin{tikzcd} 0 \arrow[d, dash] \\ C_2 \end{tikzcd} , \begin{tikzcd} 0 \arrow[d, dash] \\ C_2 \end{tikzcd} \qquad B_4 \to \begin{tikzcd} 0 \arrow[d, dash] \\ A_1 \end{tikzcd} , \begin{tikzcd} 4 \arrow[d, dash] \\ A_1 \end{tikzcd}$$

$$B_5 \to \begin{tikzcd} 0 \arrow[d, dash] \\ A_1 \end{tikzcd} , \begin{tikzcd} 5 \arrow[d, dash] \\ A_1 \end{tikzcd} \qquad B_6 \to \begin{tikzcd} 0 \arrow[d, dash] \\ A_1 \end{tikzcd} , \begin{tikzcd} 6 \arrow[d, dash] \\ A_1 \end{tikzcd}$$

$$B_7 \rightarrow \begin{array}{c}0\\ |\\ A_1\end{array} , \begin{array}{c}0\\ |\\ A_1\end{array}$$

$A_1 \rightarrow 0 , 0 \qquad A_2 \rightarrow 0 , 8$

$A_3 \rightarrow 0 , 1 \qquad A_4 \rightarrow 0 , 2$

$C_1 \rightarrow 0 , 3 \qquad C_2 \rightarrow 0 , 4$

Since there are 4 windows in the intersection area, there are 4 nodes in each tree and 8 nodes in each translation pair. There are 9 different possible labels for each node: {0,1,2,...,8}. Theoretically, there are 9^4 different trees and 9^8 different translation pairs. But after deleting some unrealistic cases (e.g., a tree like $\begin{array}{c}0—0\\ |\quad\;|\\ 0\quad\;5\end{array}$ or $\begin{array}{c}0—0\\ |\quad\;|\\ 0\quad\;7\end{array}$ is not reasonable, a translation pair like $< \begin{array}{c}0—0\\ |\quad\;|\\ 0\quad\;3\end{array} , \begin{array}{c}0—0\\ |\quad\;|\\ 0\quad\;0\end{array} >$ is not possible due to the assumption about speed limitation), there are 25 different tree patterns and 186 different translation pairs left. These 186 translation pairs are divided into 2 groups. No vehicle moves out of the intersection area for each translation pair of group I, while one or more vehicles have moved out of the intersection area for each translation pair of group II. Group I consists of 70 members. The 12 translation pairs given in the above inference example are part of these 70 pairs. Group II consists of 116 members. These 116 translation pairs are further classified into 12 subclasses according to the specific traffic type. These 12 subclasses are EW, ES, EN, WE, WS, WN, SE, SW, SN, NE, NW and NS, where E, W, S and N denotes east, west, south and north respectively and EW represents the subclass of translation pairs for which one vehicle has moved from the east to the west and left the intersection area, ES represents the subclass of translation pairs for which one vehicle has moved from the east to the south, etc. For example, $\left\langle \begin{array}{ccc} 0 & - & 3 \\ | & & | \\ 0 & & 0 \end{array}, \begin{array}{ccc} 0 & - & 0 \\ | & & | \\ 0 & & 0 \end{array} \right\rangle$ belongs to the subclass SN and $\left\langle \begin{array}{ccc} 0 & - & 4 \\ | & & | \\ 0 & & 0 \end{array}, \begin{array}{ccc} 0 & - & 0 \\ | & & | \\ 0 & & 0 \end{array} \right\rangle$ belongs to the subclass EN. There are some translation pairs which belong to more than one subclass. For example, $\left\langle \begin{array}{ccc} 0 & - & 3 \\ | & & | \\ 7 & & 0 \end{array}, \begin{array}{ccc} 0 & - & 0 \\ | & & | \\ 0 & & 0 \end{array} \right\rangle$ belongs to both subclasses SN and NS.

4.5 Implementation

The whole experiment of traffic image sequence analysis was set up in the laboratory of pattern processing and advanced automation at Purdue University. The set-up for data collection is shown in Figure 4.1.b. A white background with black vehicles is used for the traffic scene. Road areas and non-road area were specified. The digital picture sequence was taken directly through a TV scanner which is located above the traffic intersection. The digitization process was controlled interactively through a PDP 11/45 computer in the laboratory. Before digitization, we adjusted the relative distance and the focus of the TV scanner to obtain a reasonably clear picture on a TV monitor. Then everything kept stationary except the vehicles during the digitization of the whole image sequence. After each digitization of a picture, the vehicles were moved to new locations for the next picture.

The traffic analysis system using the proposed tree translation parsing is implemented in Fortran IV under Unix system on a PDP11/45 computer. The average time required for the analysis of one image is about 15 seconds (the actual computer time for each individual image depends on the number of vehicles in the image). About 90% of the time

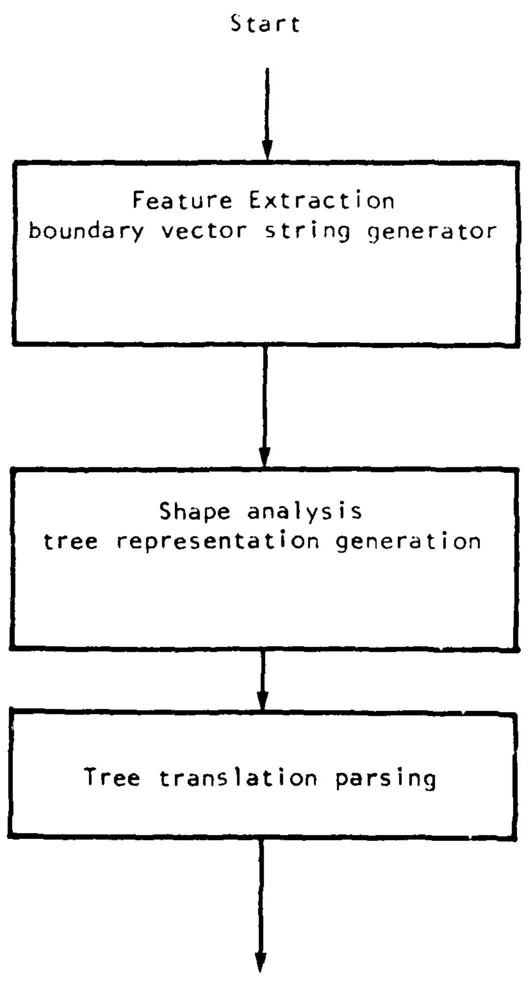

Start
Feature Extraction
boundary vector string generator
Shape analysis
tree representation generation
Tree translation parsing

Figure 4.10 Flow chart for the analysis of each image (A flow chart for tree translation parsing is given in Figure 3.1.)

is spent on the feature extraction and the shape analysis for tree node labeling.

Figure 4.11 shows the image contents of some of the image sequence. Figure 4.12 shows their corresponding tree representations around the intersection area. The correspondence between the nodes and the windowed regions is shown in Figure 4.2. The 8 nodes corresponding to the windows surrounding the intersection area are included to indicate where the vehicles come from and where they go. Figure 4.13 shows the traffic information extracted from the image sequence. This experiment illustrates an example of the application of tree translation models to the analysis of time-varying patterns. In chapter 2, we let L_1 denote the set of pattern representations of some object occurring at time t_1, L_2 denote the set of pattern represenations occurring at time t_2 and use the concept of translation to model the relation between L_1 and L_2 (or the time-varying process of the pattern representation between time t_1 and time t_2). In this traffic scene experiment, the "object" under study is the content of the intersection region in a traffic scene. L_1 and L_2 are a set of trees $\alpha: D \to \Sigma$, where D is the tree domain {0,0.1,0.2,0.2.1} and Σ is the primitive set {0,1,2,3,...,8}. Actually L_1 and L_2 are the same set. As mentioned in section 4.4, L_1 (or L_2) has 25 patterns and there are 186 possible translation pairs between L_1 and L_2. These 186 translation pairs are divided into 2 groups. Group I involves no traffic flow out of the intersection area. Group II involves traffic flow and is further classified into 12 subclasses. At each time instant $t=t_i$, the tree representations of the (i-1)th and i-th images are parsed to decide the membership. If they belong to group I, then there is no traffic flow out of the intersection area and no information update is required. If they belong to group II, then an action for the specific subclass is taken: update the traffic information for the specific path. For example, the tree representations of the intersection area for image SC01 and image SC02 of Figure 4.11 are $\begin{array}{ccc} 0 & — & 5 \\ | & & | \\ 0 & & 0 \end{array}$ and $\begin{array}{ccc} 5 & — & 0 \\ | & & | \\ 1 & & 0 \end{array}$ which are generated by the following translation rules:

$$S1 \rightarrow \begin{array}{c} 0 \\ / \ \backslash \\ A3 \quad B12 \end{array} , \begin{array}{c} 5 \\ / \ \backslash \\ A3 \quad B12 \end{array} \qquad B12 \rightarrow \begin{array}{c} 5 \\ | \\ C3 \end{array} , \begin{array}{c} 0 \\ | \\ C3 \end{array}$$

$$A3 \rightarrow 0 \ , \ 1 \qquad C3 \rightarrow 0 \ , \ 0$$

They belong to group I. Therefore no action is taken and the traffic information is unchanged. At the next time instant, the tree representations for image SC02 and image SC13 of Figure 4.10 are $\begin{array}{ccc} 5 & — & 0 \\ | & & | \\ 1 & & 0 \end{array}$ and $\begin{array}{ccc} 0 & — & 0 \\ | & & | \\ 0 & & 1 \end{array}$, which are generated by the following translation rules:

$$S2 \rightarrow \begin{array}{c} 5 \\ / \ \backslash \\ A12 \quad B17 \end{array} , \begin{array}{c} 0 \\ / \ \backslash \\ A12 \quad B17 \end{array} \qquad B17 \rightarrow \begin{array}{c} 0 \\ | \\ C13 \end{array} , \begin{array}{c} 0 \\ | \\ C13 \end{array}$$

$$A12 \rightarrow 1 \ , \ 0 \qquad C13 \rightarrow 0 \ , \ 1$$

They belong to the subclass EW of group II. Therefore the traffic information is updated and the number of vehicles moving from the east to the west is incremented by one. (In case that a translation pair belongs to two subclasses as mentioned in section 4.4, then it is required to update the traffic information of the paths specified for both subclasses.) The image sequence of this experiment does not have any significance of physical meaning. It is only used to show how the analysis system is applied. (Since there are 186 possible translation pairs (in other words, 186 time-varying patterns between consecutive images), in order to go through all of the time-varying patterns, the required number of images is far greater than 186. It is similar to the input sequence required to test every state transition of a finite-state machine.) This experiment shows a form of pattern recognition system under time-varying situation. It also demonstrates one way to extract information from an image sequence through a series of pattern translation operations. On the other hand, although the testing image sequence is successfully analyzed, further improvement of the system is still required. Similar to the cases of Chow and Aggarwal [43] and Aggarwal and Duda [42], this system assumes a fairly simple image condition for preprocessing. Therefore, to make the system working under noisy conditions, more sophisticated feature extraction technique will certainly be required.

4.6 Conclusions and Discussions

While many traffic scene-related research activities [23,68,69,80] concentrate on segmentation techniques, the proposed traffic analysis system emphasizes the representation of vehicle motion and assumes that there is little difficulty in segmentation, which is the main assumption of the system. The advantages of the proposed system include: (1) each moving object (vehicle) is allowed to have movement ranging from 0 to the length of the smallest vehicle between consecutive images, (2) the matching process is performed through a tree translation parsing which is very efficient in processing, (3) there is no need to do sophisticated prediction using information of the past history, (4) there is no need to keep the past information except the tree representation of the last image, and (5) the ability to describe an image scene and to model an image sequence. The experiment conducted here is a 2-lane traffic. When the number of traffic lanes increases, the required analysis system is essentially the same except that more windows are required and the corresponding tree representation contains more nodes. Figure 4.14 shows a 4-lane example. There are 16 windows in the intersection area. The scene is represented as a tree with 16 nodes.

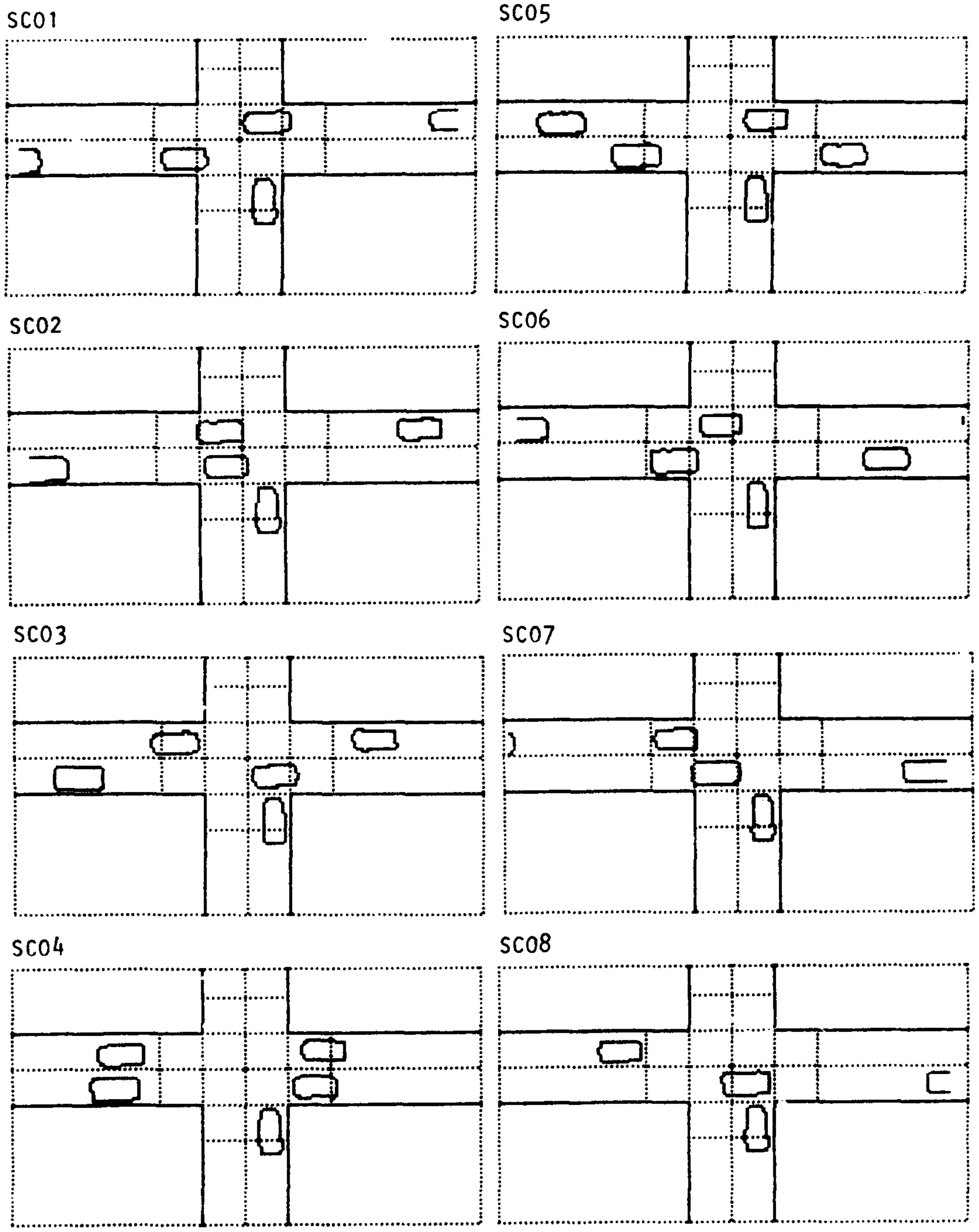


Figure 4.11 Image contents of the first 8 images in the testing image sequence

```
SC01
      0   0
      |   |
  0---0---5---0
      |   |
  1---0   0---0
      |   |
      0   3

SC02
      0   0
      |   |
  0---5---0---0
      |   |
  0---1   0---0
      |   |
      0   3

SC03
      0   0
      |   |
  5---0---0---0
      |   |
  0---0   1---0
      |   |
      0   3

SC04
      0   0
      |   |
  0---0---0---5
      |   |
  0---0   0---1
      |   |
      0   3

SC05
      0   0
      |   |
  0---0---5---0
      |   |
  0---0   0---0
      |   |
      0   3

SC06
      0   0
      |   |
  0---5---0---0
      |   |
  1---0   0---0
      |   |
      0   3

SC07
      0   0
      |   |
  5---0---0---0
      |   |
  0---1   0---0
      |   |
      0   3

SC08
      0   0
      |   |
  0---0---0---0
      |   |
  0---0   1---0
      |   |
      0   3
```

Figure 4.12 Tree representations for the images of Figure 4.11.

I.N.	EW	ES	EN	WE	WS	WN	SE	SW	SN	NE	NW	NS
2	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0
4	1	0	0	1	0	0	0	0	0	0	0	0
5	1	0	0	1	0	0	0	0	0	0	0	0
6	1	0	0	1	0	0	0	0	0	0	0	0
7	2	0	0	1	0	0	0	0	0	0	0	0
8	2	0	0	1	0	0	0	0	0	0	0	0
9	2	0	0	2	0	0	0	0	0	0	0	0
10	2	0	0	2	0	0	0	0	0	0	0	0
11	2	0	0	2	0	0	0	0	0	0	0	0
12	2	0	0	2	0	0	0	0	0	0	0	0
13	2	0	0	2	0	0	0	0	0	0	0	0
14	2	0	0	2	0	0	0	0	0	0	0	0
15	2	0	0	2	0	0	0	0	0	0	0	0
16	2	0	0	2	0	0	0	0	0	0	0	0
17	2	0	0	2	0	0	0	0	0	0	0	0
18	2	0	0	2	0	0	0	0	0	0	0	0
19	2	0	1	2	0	0	0	0	0	0	0	0
20	2	0	1	2	0	0	0	0	0	0	0	0
21	2	0	1	2	0	0	0	0	0	0	0	0
22	2	0	1	2	0	0	0	0	0	0	0	0
23	2	0	1	2	0	0	0	0	0	0	0	0
24	2	0	1	2	0	0	0	1	0	0	0	0

I.N.---image number
EW-----no. of vehicles going from east to west
ES-----no. of vehicles going from east to south
EN-----no. of vehicles going from east to north
WE-----no. of vehicles going from west to east
WS-----no. of vehicles going from west to south
WN-----no. of vehicles going from west to north
SE-----no. of vehicles going from south to east
SW-----no. of vehicles going from south to west
SN-----no. of vehicles going from south to north
NE-----no. of vehicles going from north to east
NW-----no. of vehicles going from north to west
NS-----no. of vehicles going from north to south

Figure 4.13 Traffic information extracted from the image sequence

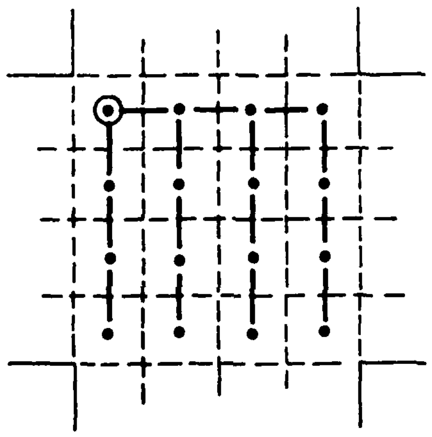


Figure 4.14 A 4-lane intersection area being divided into 16 windows and its tree representation (with the encircled node as the root)

CHAPTER 5

CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

5.1 Conclusions

A syntactic method for the analysis of time-varying image patterns is proposed and studied. This method utilizes a translation schema to model time-varying properties. A syntactic deformation model is first applied to transform the i-th image into the (i+1)-th image of an image sequence. Then the concept of translation in formal language theory is used as a mechanism to characterize the dynamic process of the sequence. In order to analyze high-dimensional patterns, the string translation schema is extended to that for trees. Formulation of stochastic translation is also presented for the modeling of stochastic properties of time-varying patterns.

A traffic scene analysis problem is analyzed using the proposed method. Each input image is presented as a tree structure. Tree translation is used to model the variation of image content between consecutive images. A parsing algorithm for the tree translation is applied to match vehicles in each pair of consecutive images. The advantages of this system are: (1) Each moving object is allowed to have larger flexible movement between consecutive images, (2) the matching process is performed through a tree translation parsing which is very efficient in processing, (3) there is no need to do sophisticated prediction using information of the past history, (4) there is no need to store the past information except the last image, and (5) has the potential for simulation of the time-varying process. This system assumes little difficulty in the segmentation of moving object, which is its main limitation.

5.2 Suggestions for Further Research

The problem of time-varying pattern analysis is a new research topic and is getting increasing attention. Although the results from the proposed syntactic method appear to be quite satisfactory, there are still some topics requiring further investigations.

(1) The translation models studied in this report include string translation and tree translation models. There are some other patterns which are better being represented as graphs [10,96,98,99]. Recently Jacobus, Chien and Selander analyze the similarities in the structure of abstract forms produced from stereoscopic motion picture sequences using a graph matching technique [72]. A pair of images contains the scenes of a 3-D object before and after the movement. The informtion concerning the boundary and surface of the object at each image is represented as a graph. A graph matching method is used to match the corresponding parts of the two graphs (assuming small overall movement). In order to make graph representation applicable for general time-varying patterns, it would be desirable to study graph language translation and stochastic graph translation.

(2) You and Fu [13] study the syntactic shape recognition using attributed grammars with promising results. Tsai and Fu [14,96] report an attributed pattern deformation model and find that sometimes it is

more powerful tc use attributed pr mitives than discrete primitives. In order to take attributes into consideration in time-varying pattern problems, one way is to extend the translation models to translation models with attributes: attributed string translation, attributed tree translation and attributed graph translation.

(3) Tree translation is applied to analyze traffic image sequence in Chapter 4. In order to further understand the usefulness of tree translation, it would be desirable to study traffic scenes using tree translation through an extensive simulation experiment.

(4) The problem of cardiac motion analysis is getting more attentions [91,92,93,94,95]. The use of computerized tomography technology is also becoming more popular. It would be interesting to study the applicability of the translation models described in this research to the problem of heart wall motion representation, or more generally, to the problem of representation of data extracted from computerized tomography.

LIST OF REFERENCES

LIST OF REFERENCES

[1] K. S. Fu, Sequential Methods in Pattern Recognition and Machine Learning. Academic Press, New York, 1968.

[2] K. Fukunaga, Introduction to Statistical Pattern Recognition. Academic Press, New York, 1972.

[3] E. A. Patrick, Fundamentals of Pattern Recognition. Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[4] H. C. Andrews, Introduction to Mathematical Techniques in Pattern Recognition, Wiley, New York, 1972.

[5] R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis. Wiley, New York, 1973.

[6] C. H. Chen, Statistical Pattern Recognition. Hayden, Washington, D.C., 1973.

[7] K. S. Fu and P. H. Swain, On Syntactic Pattern Recognition, In Software Engineering (J. T. Tou, ed.), Vol. 2, Academic Press, New York, 1971.

[8] W. F. Miller and A. C. Shaw, Linguistic Methods in Picture Processing-a Survey. Proc. AFIPS Fall Joint Comput. Conf., San Francisco, 1968, pp. 279-290.

[9] Pattern Recognition 3 (1971);4 (1972). Special issues on syntactic pattern recognition.

[10] K. S. Fu, Syntactic Methods in Pattern Recognition. New York, Academic Press, 1974.

[11] S. Y. Lu and K. S. Fu, Stochastic error-correcting syntax analysis for recognition of noisy patterns. IEEE Trans. on Computers, Vol. C-26, No. 12, Dec. 1977.

[12] S. Y. Lu and K. S. Fu, Error-correcting tree automata for syntactic pattern recognition. IEEE Trans. on Computers, Vol. C-27, Nov. 1978.

[13] K. C. You and K. S. Fu, Syntactic shape recognition using attributed grammar. Technical Report TR-EE 78-38, Purdue University.

[14] W. H. Tsai and K. S. Fu, A pattern deformation model and bayes error-correcting recognition system. *Technical Report* TR-EE 78-26, Purdue University.

[15] A. A. Arking and R. C. Lo and A. Rosenfeld, An evaluation of fourier transform techniques for cloud motion estimation, Computer Science Technique Report TR-351, University of Maryland, Jan. 1975.

[16] Abstracts of the workshop on computer analysis of time-varying imagery. April 1979.

[17] H. Nagel, Analysis techniques for image sequences, 1978 International Joint Conference on Pattern Recognition.

[18] J. A. Leese, C. S. Novak and V. R. Taylor, The determination of cloud pattern motions from geosynchronous satellite image data, Pattern Recognition, 2, 1970, 279-292.

[19] R. P. Futrelle and G. C. Speckert, Extraction of motion data by interactive image processing, 1978 Proc. of Pattern Recognition and Image Processing Conference, 405-408, Chicago, Illinois.

[20] W. N. Martin and J. K. Aggarwal, Survey: Dynamic scene analysis. Computer Graphics Image Processing, 7, 1978, 356-374.

[21] L. F. Hubert and L. F. Whitney, Wind estimation from geostationary satellite pictures. Monthly Weather Review, 99, 1971, 665-672.

[22] K. Wolferts, Special problems in interactive image processing for traffic analysis. 1974 International Joint Conference on Pattern Recognition, 1-2.

[23] M. Onoe and K. Ohba, Digital image analysis of traffic flow. 1976 International Joint Conference on Pattern Recognition, 803-808.

[24] R. T. Chien and V. C. Jones, Acquisition of moving objects and hand-eye coordination. 1975 International Joint Conference on Artificial Intelligence, 737-741.

[25] R. Jain and H. H. Nagel, On a motion analysis process for image sequences from real world scenes. IEEE workshop on pattern recognition and artificial intelligence, Princeton, N.J., 1978.

[26] C. A. Rosen and D. Nitzan, Use of sensors in programmable automation. Computer, 10, Dec. 1977, 12-23.

[27] B. Neumann, Interpretation of imperfect object contours. 1978 International Joint Conference on Pattern Recognition.

[28] T. Uno, M. Ejiri and T. Kogunaga, A method of real-time recognition of moving objects and its application. Pattern Recognition, 8, (1976), 201-208.

[29] R. Eskenazi and R. T. Cunningham, Real-time tracking of moving objects in TV images. IEEE workshop on Pattern recognition and artificial intelligence, Princeton, N.J., 1978.

[30] C. K. Chow and T. Kaneko, Automatic boundary detection of the left ventricle from cineangiograms. Computers and Bimedical Research 5 (1972) 388-410.

[31] C. K. Chow, S. K. Hilal and K. E. Nigbuhr, X-ray image subtraction by digital means. IBM J. Res. Develop. 17 (1973) 206-218.

[32] K. Hachimura, M. Kuwahara and M. Knoshita, Left ventricular contour extraction from radioisotope angiocardiograms and classification of left ventricular wall motion. 1978 International Joint Conference on Pattern Recognition.

[33] S. A. Johnson et al., Bioimage synthesis and analysis from x-ray, gamma, optical and ultrasound energy. Digital Processing of Biomedical Images, (K. Preston and M. Onoe eds.) 1976, 203-226.

[34] G. T. Herman and H. K. Liu, Dynamic boundary surface detection. Computer Graphics Image Processing 7, 1978, 130-138.

[35] H. K. Liu, Two- and three-dimensional boundary detection. Computer Graphics Image Processing 6, 1977, 123-134.

[36] P. H. Heintzen et al. Automated video-angiocardiographic image analysis. Computer 8, 1975, 55-64.

[37] M. Yachida, M. Asada and S. Tsuji, Automatic motion analysis system of moving objects from the records of natural processes. 1978 International Joint Conference on Pattern Recognition.

[38] D. Davenport et al. The investigation of the behavior of microorganisms by computerized television. IEEE Trans. on Biomedical Eng. BME-17 (1970) 230-237.

[39] Y. Ariki, T. Kanade and T. Sakai, An interactive image modeling and tracing system. 1978 International Joint Conference on Pattern Recognition.

[40] M. Takagi and K. Sakaue, The analysis of moving granules in a pancreatic cell by digital moving image processing. 1978 International Joint Conference on Pattern Recognition.

[41] R. C. Lo, and J. A. Parikh, A study of the application of Fourier transforms to cloud movement estimation from satellite photographs, Computer Science Technical Report TR-242, University of Maryland, 1973.

[42] J. K. Aggarwal and R. O. Duda, Computer analysis of moving polygonal images, IEEE Trans. Computers, C-24, Oct. 1975, 966-976.

[43] W. K. Chow and J. K. Aggarwal, Computer analysis of planar curvilinear moving images, IEEE Trans. Computers C-26, 1977, 179-185.

[44] J. L. Potter, Scene segmentation by velocity measurements obtained with a cross-shaped template, 4IJCAI, pp. 803-808, Tbilisi, Georgia, U.S.S.R., Sept. 1975.

[45] R. Nevatia, Depth measurement by motion stereo, Computer Graphics Image Processing, 5, 1976, 203-214.

[46] M. S. Ulstad, An algorithm for estimating small scale differences between two digital images, Pattern Recognition, 5, 1973, 323-333.

[47] R. L. Lillestrand, Techniques for change detection, IEEE Trans. Computers C-21, 1972, 654-659.

[48] G. G. Pieroni, A method for analyzing dynamic processes represented by chains of maps.

[49] Y. Ariki, A system for analyzing time-varying image patterns,

[50] O. T. vonRamm, F. L. Thurstone, Cardiac imaging using a phased array ultrasound system. I. System Design. Circulation. Vol. 53, No. 2, pp. 258-261, 1976.

[51] A. V. Aho and J. D. Ullman, The Theory of Parsing, Translation, and Compiling, Vol. 1, Englewood Cliffs, N.J.: Prentice-Hall, 1972.

[52] L. W. Fung and K. S. Fu, "Stochastic syntactic decoding for pattern classification," IEEE Trans. on Comput., Vol. C-24, No. 6, July 1975.

[53] D. J. Rosenkrantz, programmed grammars--a new device for generating formal languages, Ph.D. dissertation, Columbia Univ., New York (1967).

[54] P. H. Swain and K. S. Fu, "Stochastic programmed grammars for syntactic pattern recognition," Pattern Recognition, Vol. 4, 1972.

[55] K. S. Fu and B. K. Bhargava, "Tree systems for syntactic pattern recognition," IEEE Trans. on Comput., Vol. C-22, No. 12, Dec. 1973.

[56] S. W. Zucker, "Toward a model of texture," Computer Graphics and Image Processing, 5, 1976, pp. 190-202.

[57] A. V. Aho and J. D. Ullman, Translations on a context-free grammar, Inform. Contr. 19, 1971, 439-475.

[58] B. Baker, Generalized syntax directed translation, tree transducers, and linear space, SIAM J. on Computing, Vol. 7, No. 3, Aug. 1978.

[59] W. Brainerd, Tree generating regular systems, Information and Control, 14, 1969, pp. 217-231.

[60] H. C. Lee and K.-S. Fu, "A stochastic syntax analysis procedure and its application to pattern classification," IEEE Trans. Comput., Vol. C-21, July 1972, pp. 660-666.

[61] B. Moayer and K. S. Fu, "A tree system approach for fingerpring pattern recognition," IEEE Trans. on Comput., Vol. C-25, No. 3, March 1976.

[62] S. Y. Lu and K. S. Fu, A syntactic approach to texture analysis, Computer Graphics and Image Processing 7, 1978, 303-330.

[63] S. Y. Lu and K. S. Fu, Stochastic tree grammar inference for texture synthesis on discrimination, 1978 Proc. of IEEE Pattern Recognition and Image Processing Conference, Chicago.

[64] R. C. Lo, J. Mohr and J. A. Parikh, Applications of Fourier transform methods of cloud movement estimation to simulated and satellite photographs, Computer Science Technical Report TR-292, University of Maryland, 1974.

[65] R. C. Lo and J. Johr, Application of enhancement and thresholding techniques to Fourier transform cloud motion estimates. Computer Science Technical Report TR-326, University of Marlyland, 1974.

[66] E. A. Smith and D. R. Phillips, Automated cloud tracking using precisely aligned digital ATS pictures, IEEE Trans. Computers, C-21, 1972, 715-729.

[67] J. O. Limb and J. A. Murphy, Estimating the velocity of moving images in television signals, Computer Graphics and Image Processing, 4, 1975, 311-329.

[68] H. H. Nagel, Formation of an object concept by analysis of systematic time variations in the optically perceptible environment. Computer Graphics and Image Processing, 7, 1978, 149-194.

[69] R. Jain and H. H. Nagel, On the analysis of accumulative difference pictures from image sequences of real world scenes, IEEE Trans. Pattern Analysis Machine Intelligence, Vol. PAMI-1 Apr. 1979, 206-213.

[70] R. Jain, D. Militzer, and H. H. Nagel, Separating nonstationary from stationary scene components in a sequence of real world TV images, 1977 Proc. IJCAI, Cambridge, MA, Aug. 1977.

[71] S. Ullman, The Interpretation of Visual Motion, Cambridge, MA, MIT Press, 1979.

[72] C. J. Jacobus, R. T. Chien and J. M. Selander, Motion detection and analysis by matching graphs of intermediate-level primitives, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 6, Nov. 1980.

[73] J. O'Rourke and N. I. Badler, Model-based image analysis of human motion using constraint propagation, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 6, Nov. 1980.

[74] J. K. Tsotsos, etc., A framework for visual motion understanding, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 6, Nov. 1980.

[75] J. W. Roach and J. K. Aggarwal, Determining the movement of objects from a sequence of images, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 6, Nov. 1980.

[76] M. Yachida, M. Ikeda and S. Tsuji, A plan-guided analysis of cineangiograms for measurement of dynamic behavior of heart wall, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 6, Nov. 1980.

[77] M. Yachida, M. Asada, and S. Tsuji, Automatic analysis of moving objects from the records of natural processes.

[78] S. Tsuji, M. Asada, and m. Yachida, Tracking and segmentation of moving objects in dynamic line images, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 6, Nov. 1980.

[79] J. W. Roach and J. K. Aggarwal, Computer tracking of objects moving in space, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 2, April 1979.

[80] W. B. Thompson, Combining motion and contrast for segmentation. IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 6, Nov. 1980.

[81] C. L. Fennema and W. B. Thompson, Velocity determination in scenes containing several moving objects, Computer Graphics and Image Processing, 9, 1979, 301-315.

[82] A. Z. Meiri, On monocular perception of 3-D moving objects, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 6, Nov. 1980.

[83] F. J. Maryanski and M. G. Thomason, Properties of stochastic syntax-directed translation schemata, Intl. J. of Computer and Information Science, Vol. 5, No. 2, 1979.

[84] M. G. Thomason, Stochastic syntax-directed translation schemata for correction of errors in context-free languages, IEEE Trans. Computer, C-24, 1975.

[85] K. S. Fu and T. L. Booth, Grammatical inference: introduction and survey - Part I and II, IEEE Trans. Sys. Man Cybernetics, Vol. SMC-5, No. 1 & 4, 1975.

[86] J. M. Brayer and K. S. Fu, A note on the k-tail method of tree grammar inference, IEEE Trans. System Man and Cybernetics, April 1977.

[87] Barry Levine, Derivatives of tree sets with applications to grammatical inference.

[88] S. W. Zucker, R. A. Hummel and A. Rosenfeld, An application of relaxation labeling to line and curve enhancement, IEEE Trans. Comput., Vol. C-26, Apr. 1977.

[89] W. Scacchi, Visual motion perception by intelligent systems, Proc. Pattern Recog. Image Proc., Aug. 1979.

[90] J. B. Garrison, et al., Quantifying regional wall motion and thickening in two-dimensional echocardiography with a computer-aided contouring system, 1977 Proc. Computers in Cardiology.

[91] P. D. Clayton, et al., A computer system for the cardiovascular laboratory, 1974 Proc. Computers in Cardiology.

[92] B. R. Hieb, et al., A computerized system for segmental analysis of sequential left ventricular cineangiogram frames, 1976 Proc. Computers in Cardiology.

[93] P. H. Heintzen, et al., Automated video-angiocardiographicsh image analysis, 1974 Proc. Computers in Cardiology.

[94] R. Balocchi, et al., A system for off-line analysis of left ventricular angiographic images by the use of a minicomputer, 1977 Proc. Computer in Cardiology.

[95] R. W. Brower and G. T. Meester, Computer based methods for quantifying regional left ventricular wall motion from cineventriculograms, 1976 Proc. Computers in Cardiology.

[96] W. H. Tsai and K. S. Fu, Error-correcting isomophisms of attributed relational graphs for pattern analysis, IEEE Trans. System, Man and Cybernetics, Vol. SMC-9, No. 12, Dec. 1979.

[97] T. I. Fan and K. S. Fu, A syntactic approach to time-varying image analysis, Computer Graphics Image Processing, 11, 1979.

[98] R. M. Haralick and J. S. Kartus, Arrangements, homomorphisms and discrete relaxations, IEEE Trans. System, Man and Cybernetics, Vol. SMC-8, Aug. 1978.

[99] T. Pavlidis, Gramatical and graph theoretical analysis of pictures, in Graphic languages, Nake and Rosenfeld, eds. Amsterdam, North-Holland, 1972.

[100] G. Y. Tang and T. S. Huang, A syntactic-semantic approach to image understanding and creation, IEEE Trans. Pattern Analysis Machine Intelligence, Vol. PAMI-1, No. 2, Apr. 1979.

BIBLIOGRAPHIC DATA SHEET	1. TR-EE 81-10	2. AD-A101665
4. A SYNTACTIC METHOD FOR TIME-VARYING PATTERN ANALYSIS		5. May 1981
		6.
7. Tzu-I J. Fan and K. S. Fu		8. TR-EE 81-10
9. Purdue University School of Electrical Engineering West Lafayette, IN 47907		10.
		11. N000 14-C-0574
12. Office of Naval Research		13.
		14.
15.		

16.

A syntactic method for the analysis of time-varying image patterns is proposed and studied. This method utilizes translation schema to model the time-varying properties of image patterns. A syntactic deformation model is first applied to transform the i-th image into the (i+1)-th image of an image sequence. Then the concept of translation in formal language theory is used as a mechanism to characterize the dynamic process of the image sequence. A formulation of stochastic translation is also presented. A generalized syntax-directed tree translation model is proposed to handle high-dimensional patterns. The generalized model is compared with the conventional top-down and bottom-up tree translation models.

A traffic monitoring problem is analyzed using the proposed tree translation model. Each input image is represented as a tree structure. The proposed tree translation model is used to model the variation of image content between consecutive images. A parsing algorithm for tree translation is applied to match moving objects (vehicles) in each pair of consecutive images.

17.

17b.

17c.

18. DISTRIBUTION STATEMENT A Approved for public release; Distribution Unlimited	19. UNCLASSIFIED	21. 134
	20.	22.

ATE

LMED

-8